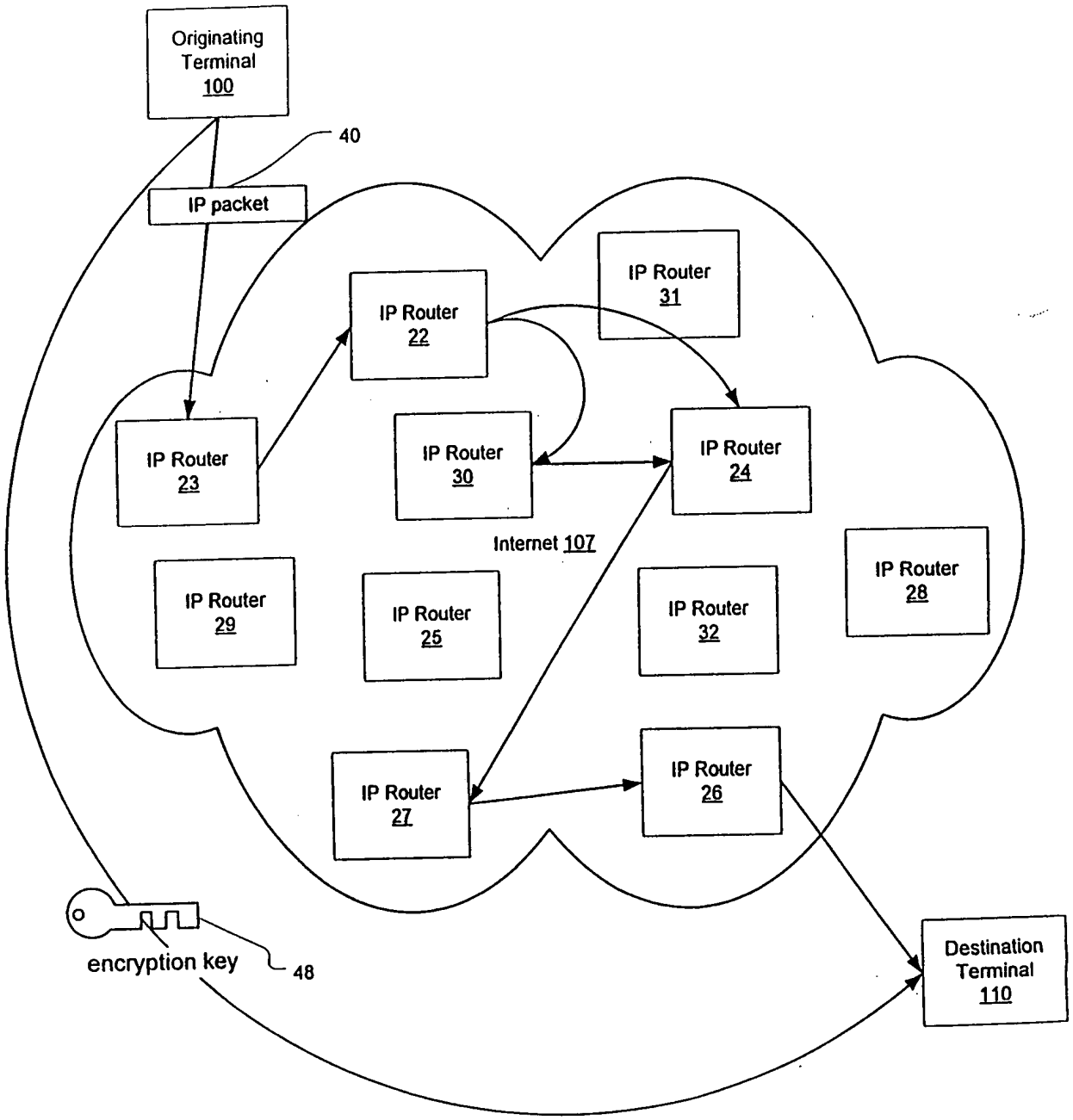

Originating Terminal 100
40
IP packet
IP Router 22
IP Router 31
IP Router 23
IP Router 30
IP Router 24
Internet 107
IP Router 29
IP Router 25
IP Router 32
IP Router 28
IP Router 27
IP Router 26
encryption key
48
Destination Terminal 110

Fig. 1

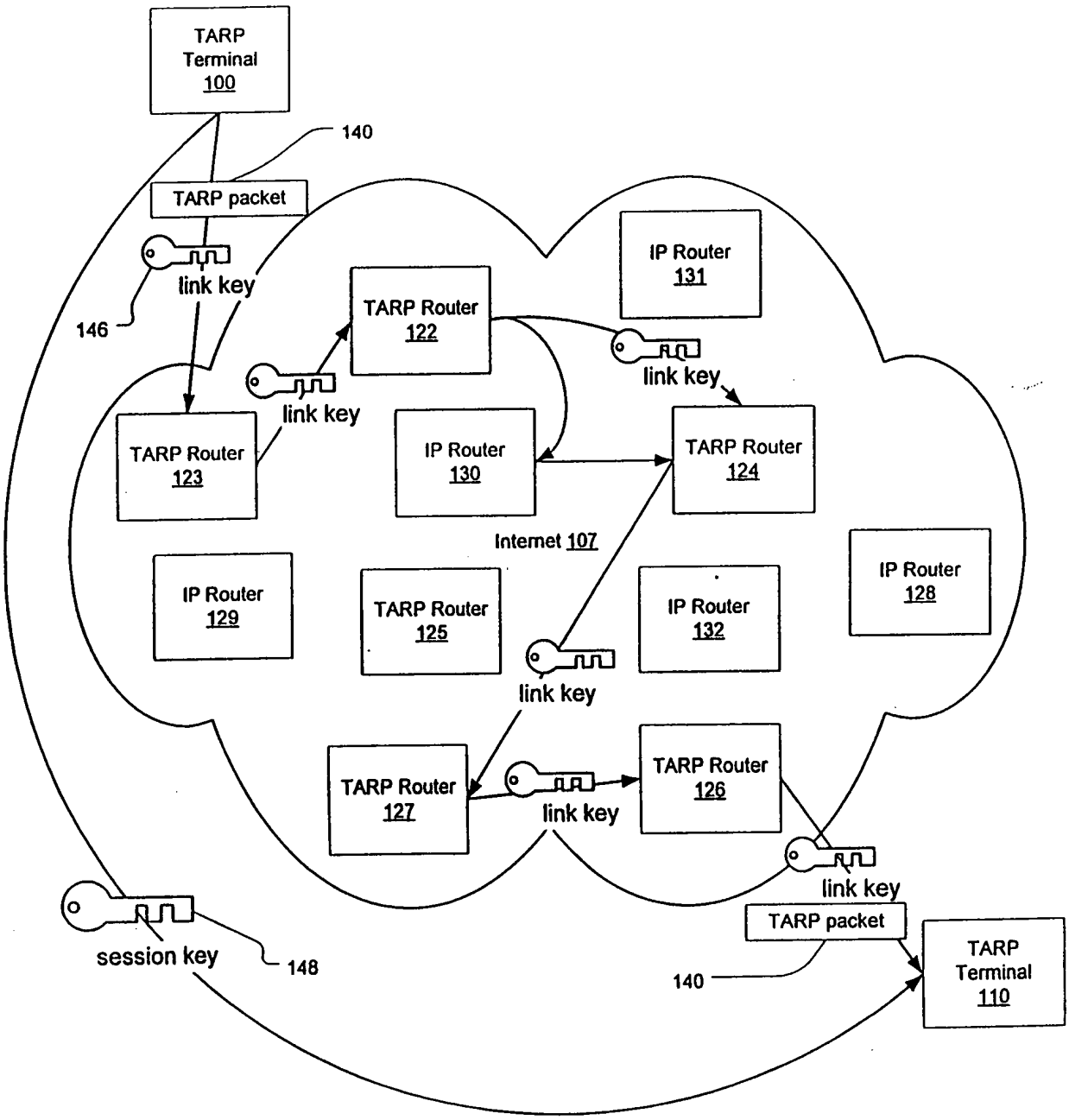

TARP Terminal 100
140
TARP packet
link key
146
TARP Router 122
link key
IP Router 131
link key
TARP Router 123
IP Router 130
TARP Router 124
Internet 107
IP Router 129
TARP Router 125
IP Router 132
IP Router 128
link key
TARP Router 127
link key
TARP Router 126
link key
TARP packet
session key
148
140
TARP Terminal 110

Fig. 2

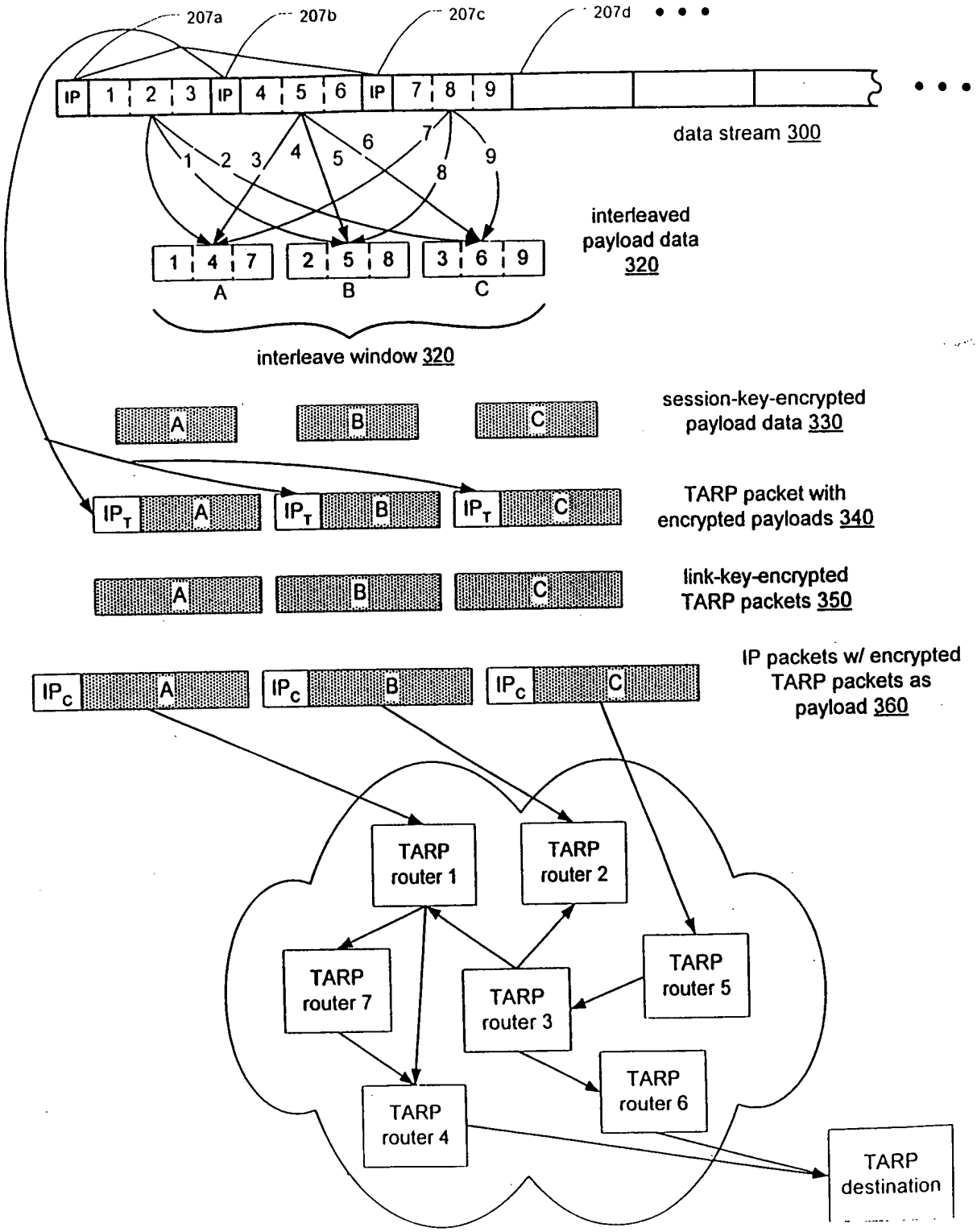

207a
207b
207c
207d
IP
1 2 3
IP
4 5 6
IP
7 8 9
data stream 300
interleaved payload data 320
1 4 7
2 5 8
3 6 9
A
B
C
interleave window 320
session-key-encrypted payload data 330
TARP packet with encrypted payloads 340
link-key-encrypted TARP packets 350
IP packets w/ encrypted TARP packets as payload 360
TARP router 1
TARP router 2
TARP router 3
TARP router 4
TARP router 5
TARP router 6
TARP router 7
TARP destination

Fig. 3a

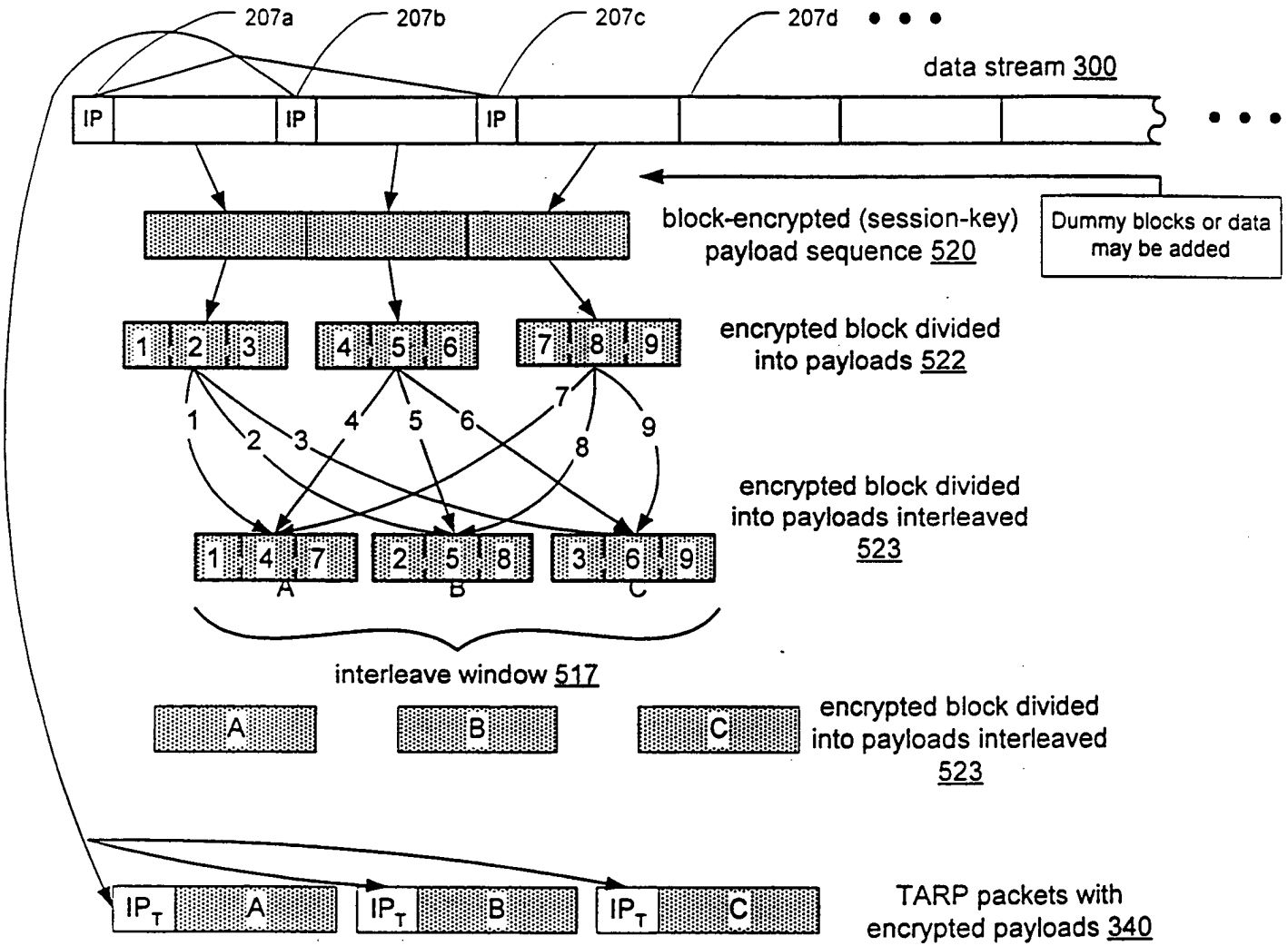

207a
207b
207c
207d
data stream 300
IP
IP
IP
block-encrypted (session-key) payload sequence 520
Dummy blocks or data may be added
1 2 3
4 5 6
7 8 9
encrypted block divided into payloads 522
1 4 7
2 5 8
3 6 9
A
B
C
encrypted block divided into payloads interleaved 523
interleave window 517
A
B
C
encrypted block divided into payloads interleaved 523
IP$_T$ A
IP$_T$ B
IP$_T$ C
TARP packets with encrypted payloads 340

Fig. 3b

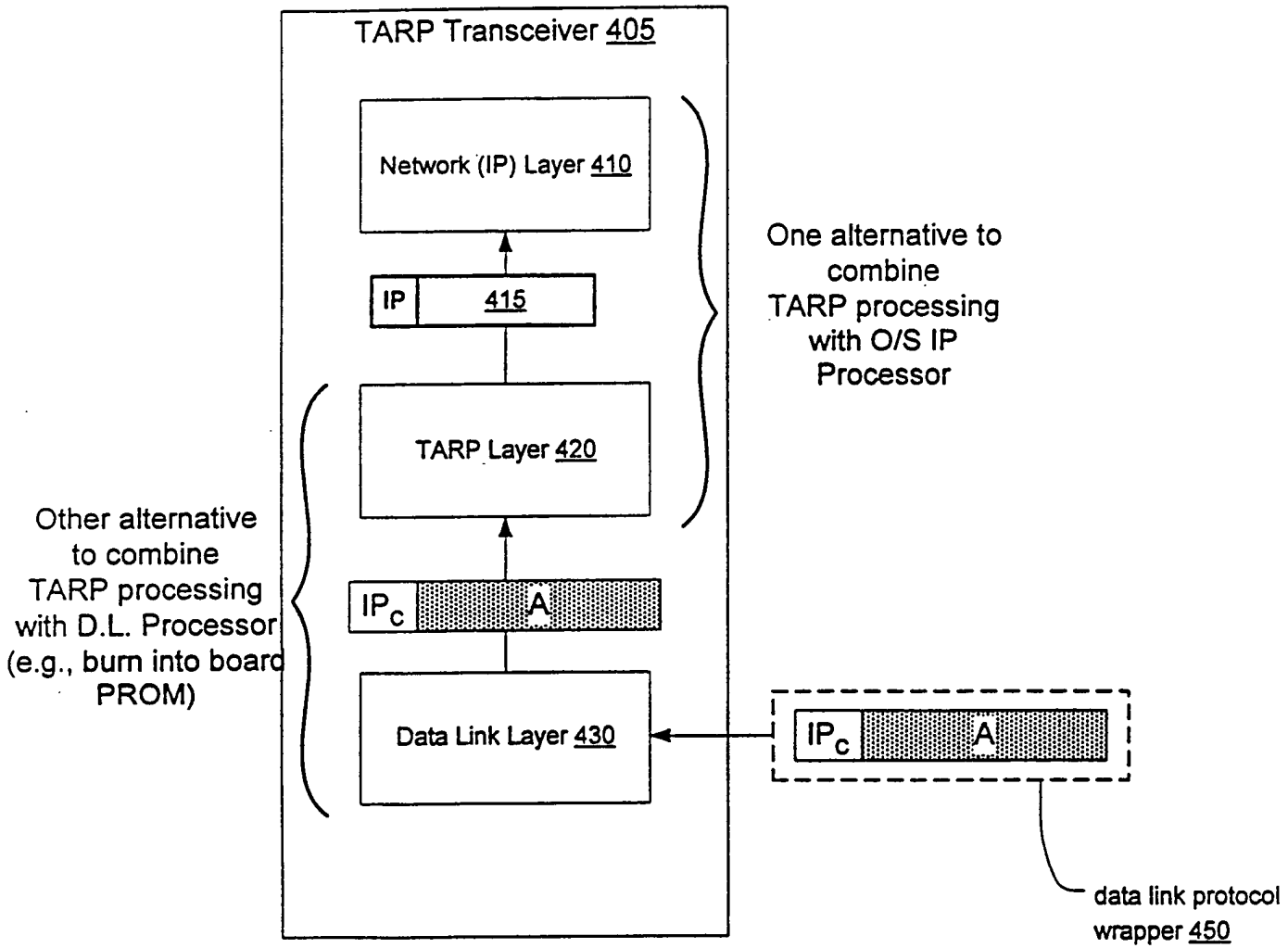

TARP Transceiver 405
Network (IP) Layer 410
IP
415
TARP Layer 420
IPc
A
Data Link Layer 430
IPc
A
One alternative to combine TARP processing with O/S IP Processor
Other alternative to combine TARP processing with D.L. Processor (e.g., burn into board PROM)
data link protocol wrapper 450

Fig. 4

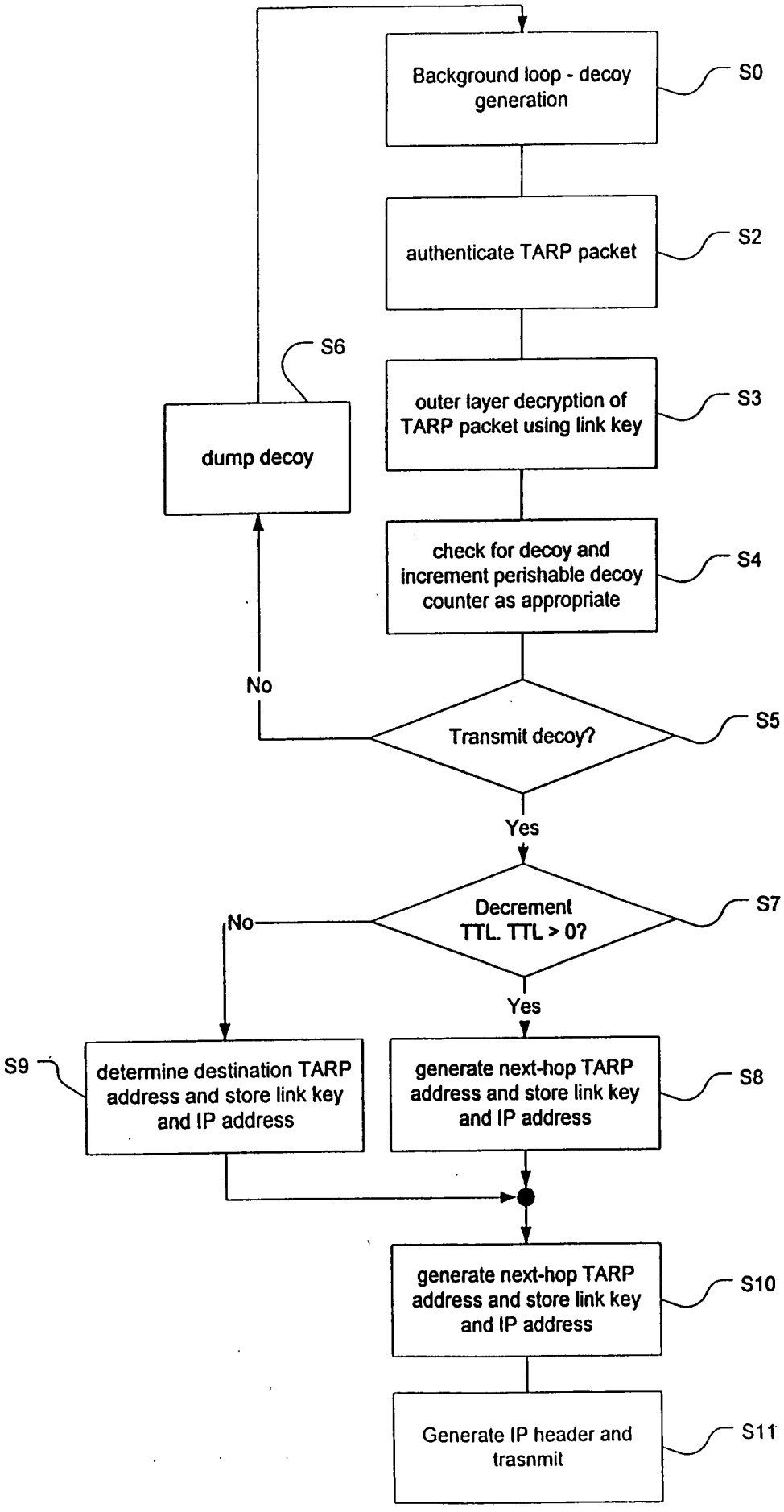

Background loop - decoy generation
S0
authenticate TARP packet
S2
outer layer decryption of TARP packet using link key
S3
check for decoy and increment perishable decoy counter as appropriate
S4
Transmit decoy?
S5
No
Yes
dump decoy
S6
Decrement TTL. TTL > 0?
S7
No
Yes
determine destination TARP address and store link key and IP address
S9
generate next-hop TARP address and store link key and IP address
S8
generate next-hop TARP address and store link key and IP address
S10
Generate IP header and trasnmit
S11

Fig. 5

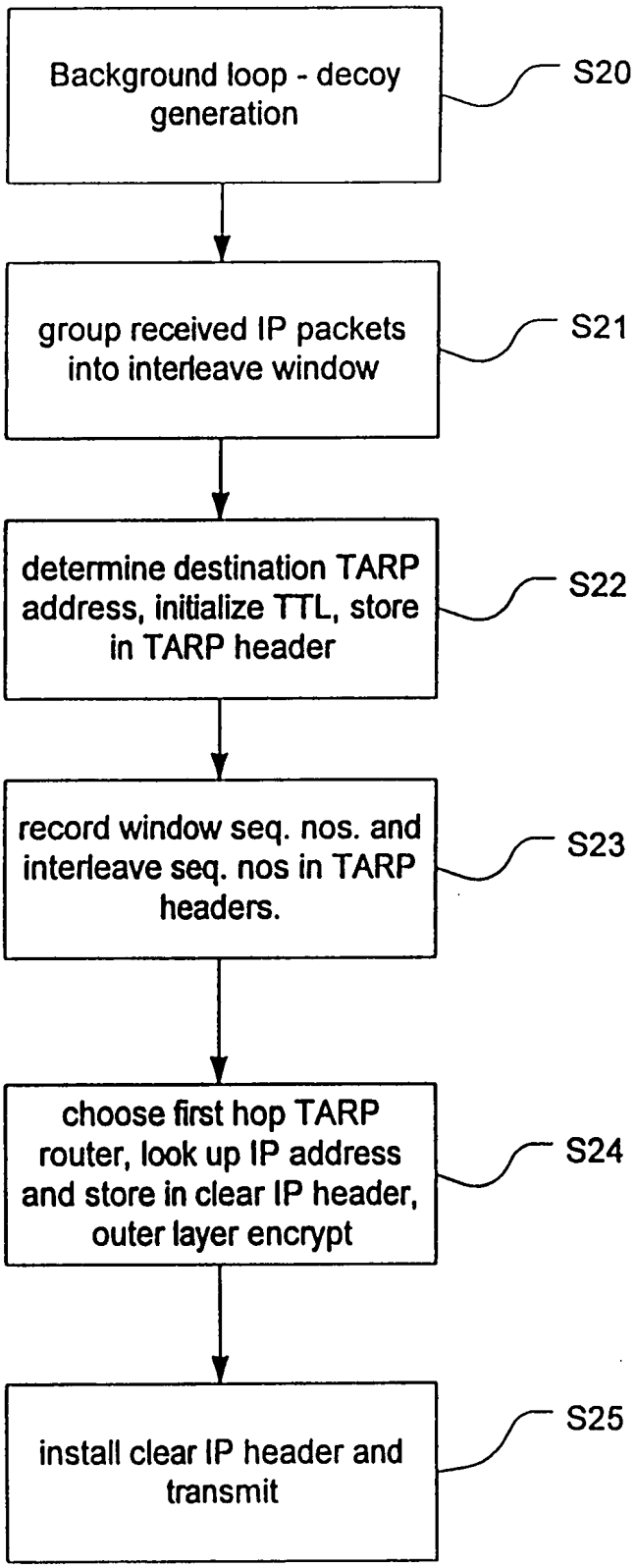

Background loop - decoy generation
S20
group received IP packets into interleave window
S21
determine destination TARP address, initialize TTL, store in TARP header
S22
record window seq. nos. and interleave seq. nos in TARP headers.
S23
choose first hop TARP router, look up IP address and store in clear IP header, outer layer encrypt
S24
install clear IP header and transmit
S25

Fig. 6

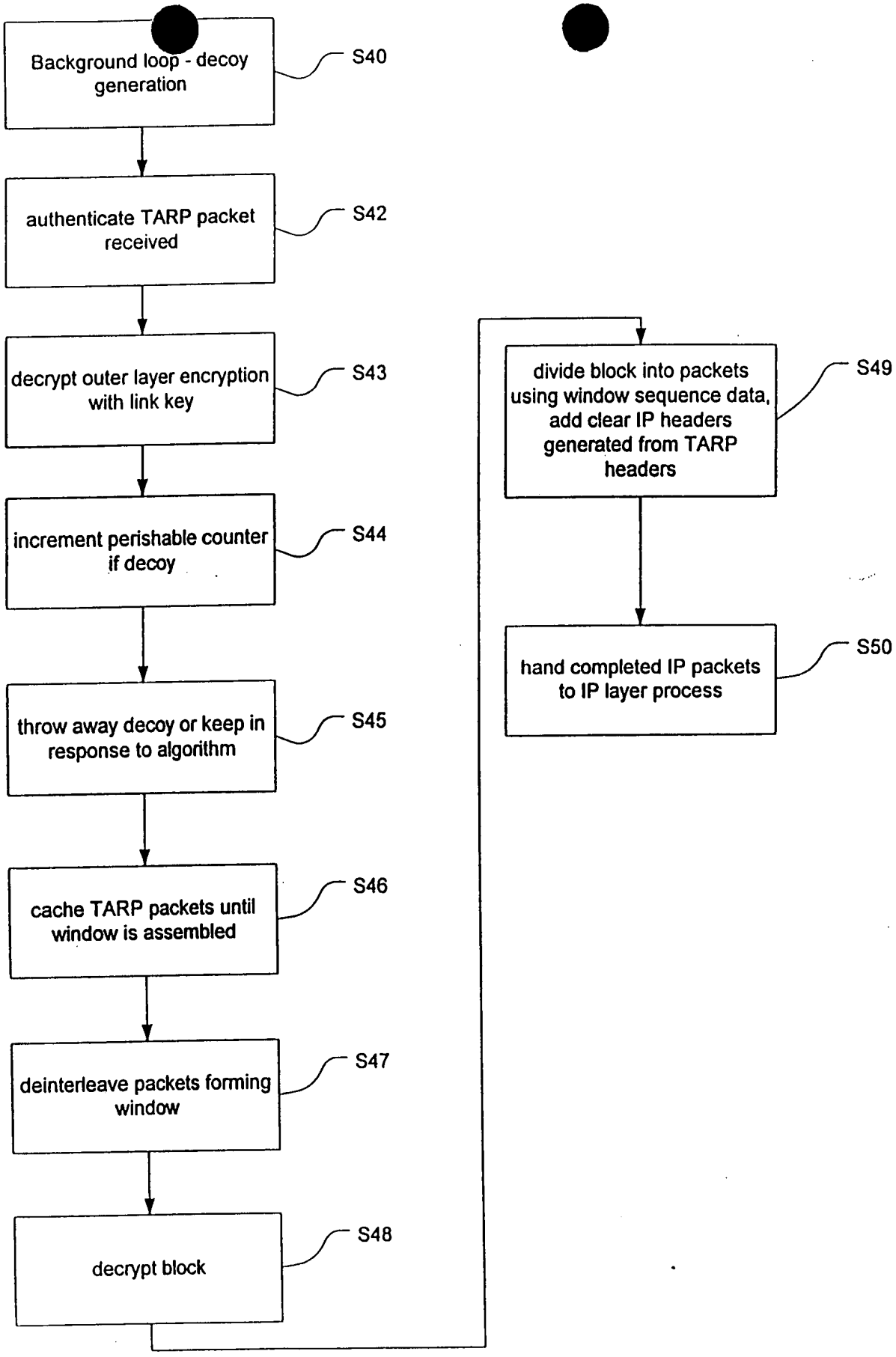

Background loop - decoy generation
S40
authenticate TARP packet received
S42
decrypt outer layer encryption with link key
S43
increment perishable counter if decoy
S44
throw away decoy or keep in response to algorithm
S45
cache TARP packets until window is assembled
S46
deinterleave packets forming window
S47
decrypt block
S48
divide block into packets using window sequence data, add clear IP headers generated from TARP headers
S49
hand completed IP packets to IP layer process
S50

Fig. 7

FIG. 8

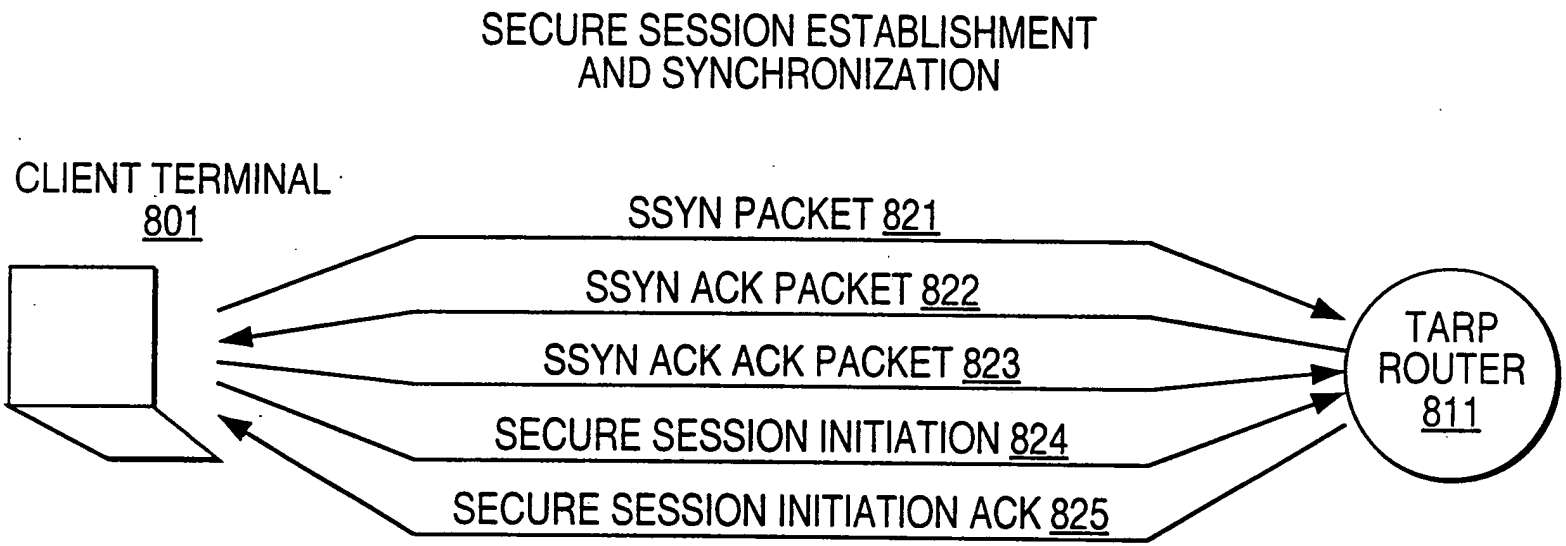
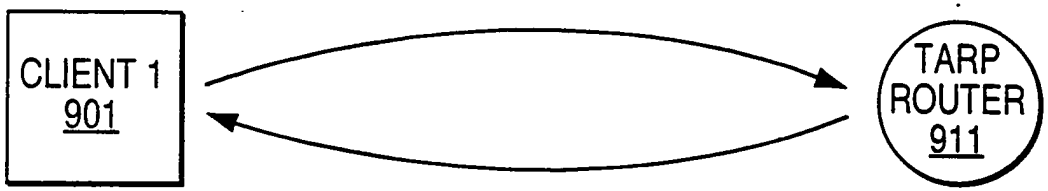


FIG. 9

IHOP TRANSMIT AND RECEIVE TABLES



TRANSMIT TABLE 921

131.218.204.98	,	131.218.204.65
131.218.204.221	,	131.218.204.97
131.218.204.139	,	131.218.204.186
131.218.204.12	,	131.218.204.55
.		.
.		.
.		.

RECEIVE TABLE 924

131.218.204.98	,	131.218.204.65
131.218.204.221	,	131.218.204.97
131.218.204.139	,	131.218.204.186
131.218.204.12	,	131.218.204.55
.		.
.		.
.		.

RECEIVE TABLE 922

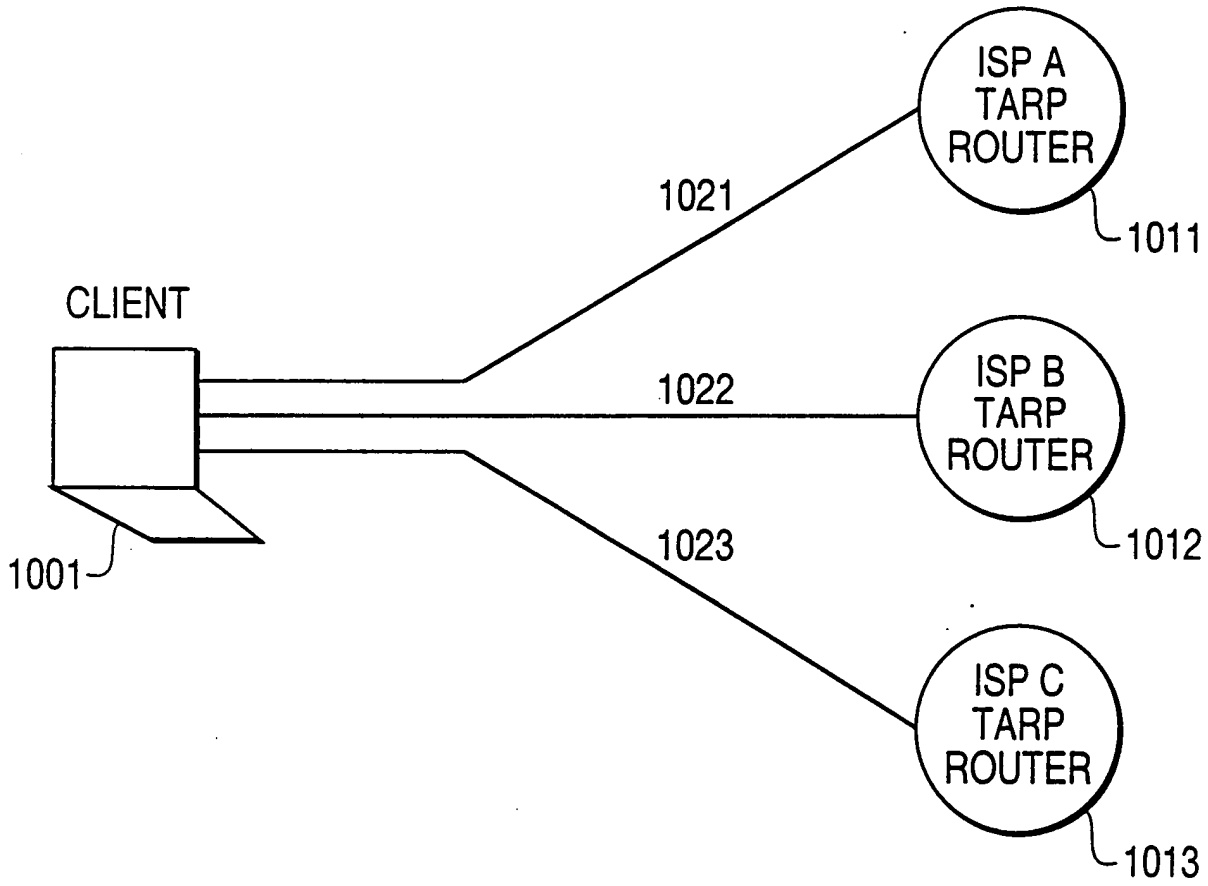
131.218.204.161	,	131.218.204.89
131.218.204.66	,	131.218.204.212
131.218.204.201	;	131.218.204.127
131.218.204.119	,	131.218.204.49
.		.
.		.
.		.

TRANSMIT TABLE 923

131.218.204.161	,	131.218.204.89
131.218.204.66	,	131.218.204.212
131.218.204.201	,	131.218.204.127
131.218.204.119	,	131.218.204.49
.		.
.		.
.		.

FIG. 10

PHYSICAL LINK REDUNDANCY



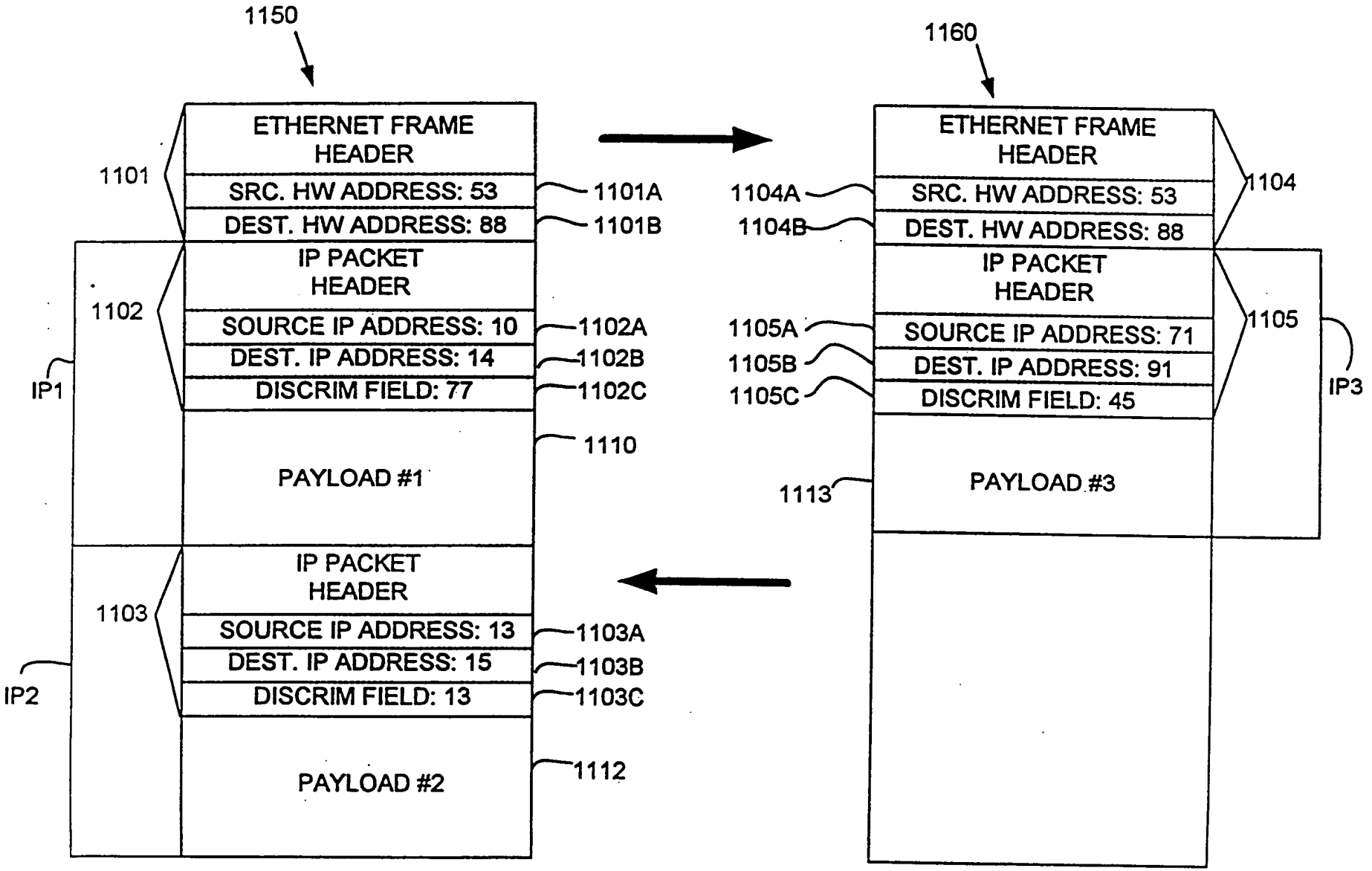

1150
1160
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1101
1101A
1101B
1102
1102A
1102B
1102C
1110
1103
1103A
1103B
1103C
1112
1104
1104A
1104B
1105
1105A
1105B
1105C
1113
IP1
IP2
IP3

FIG. 11

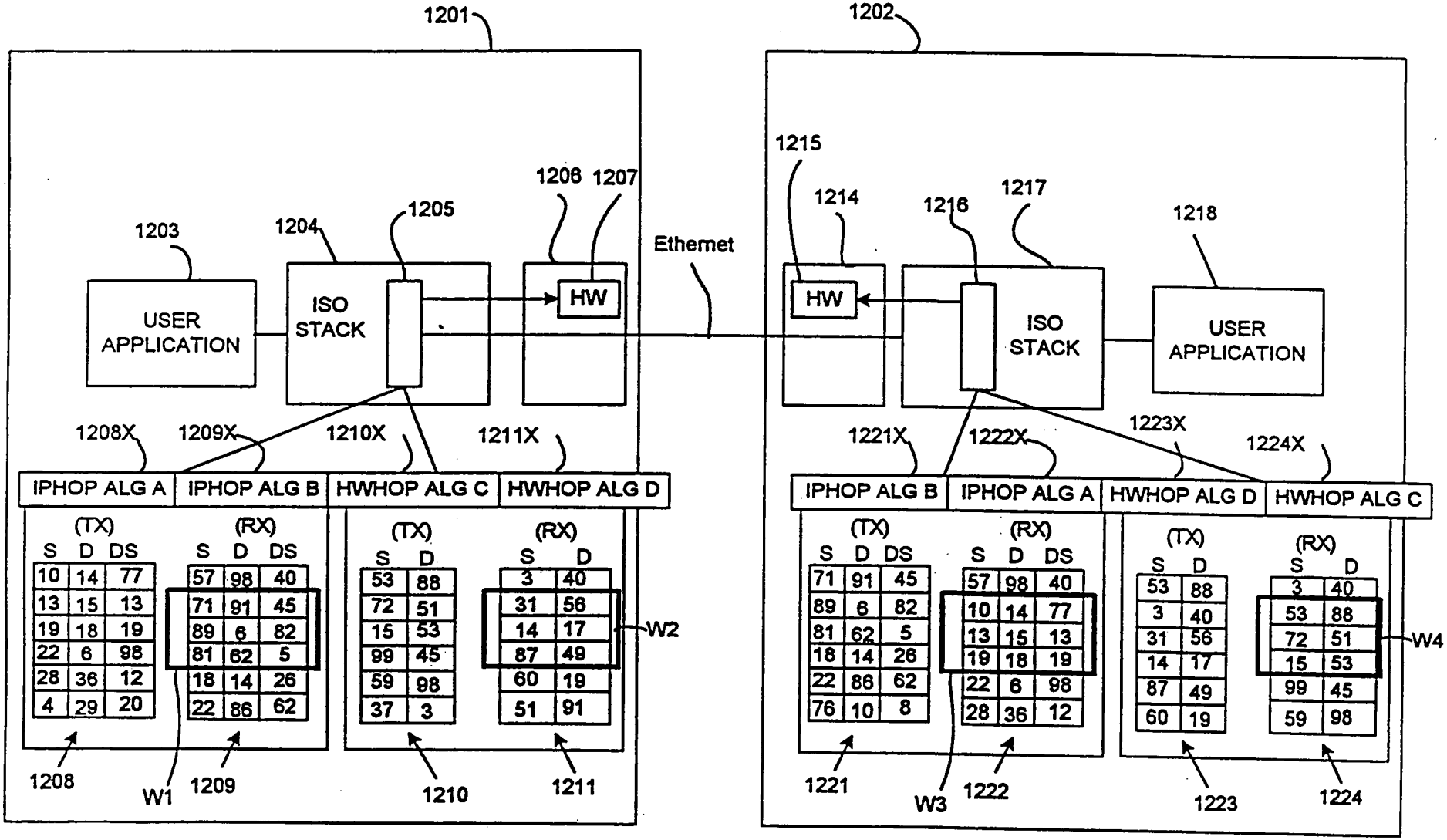

1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1214
1215
1216
1217
1218
USER APPLICATION
ISO STACK
HW
Ethernet
HW
ISO STACK
USER APPLICATION
1208X
1209X
1210X
1211X
1221X
1222X
1223X
1224X
IPHOP ALG A
IPHOP ALG B
HWHOP ALG C
HWHOP ALG D
IPHOP ALG B
IPHOP ALG A
HWHOP ALG D
HWHOP ALG C
(TX)
(RX)
S D DS
S D
W1
W2
W3
W4

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

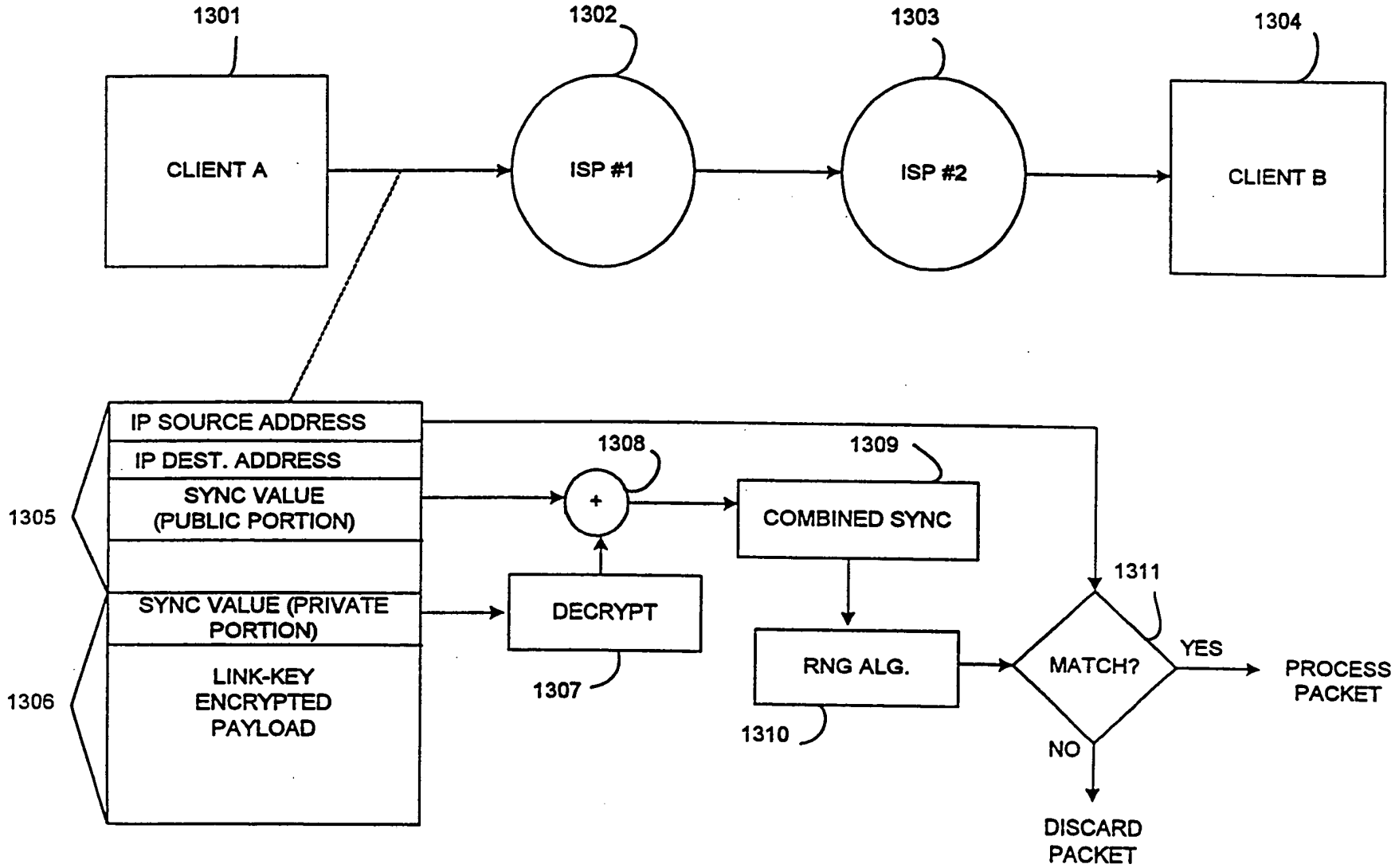
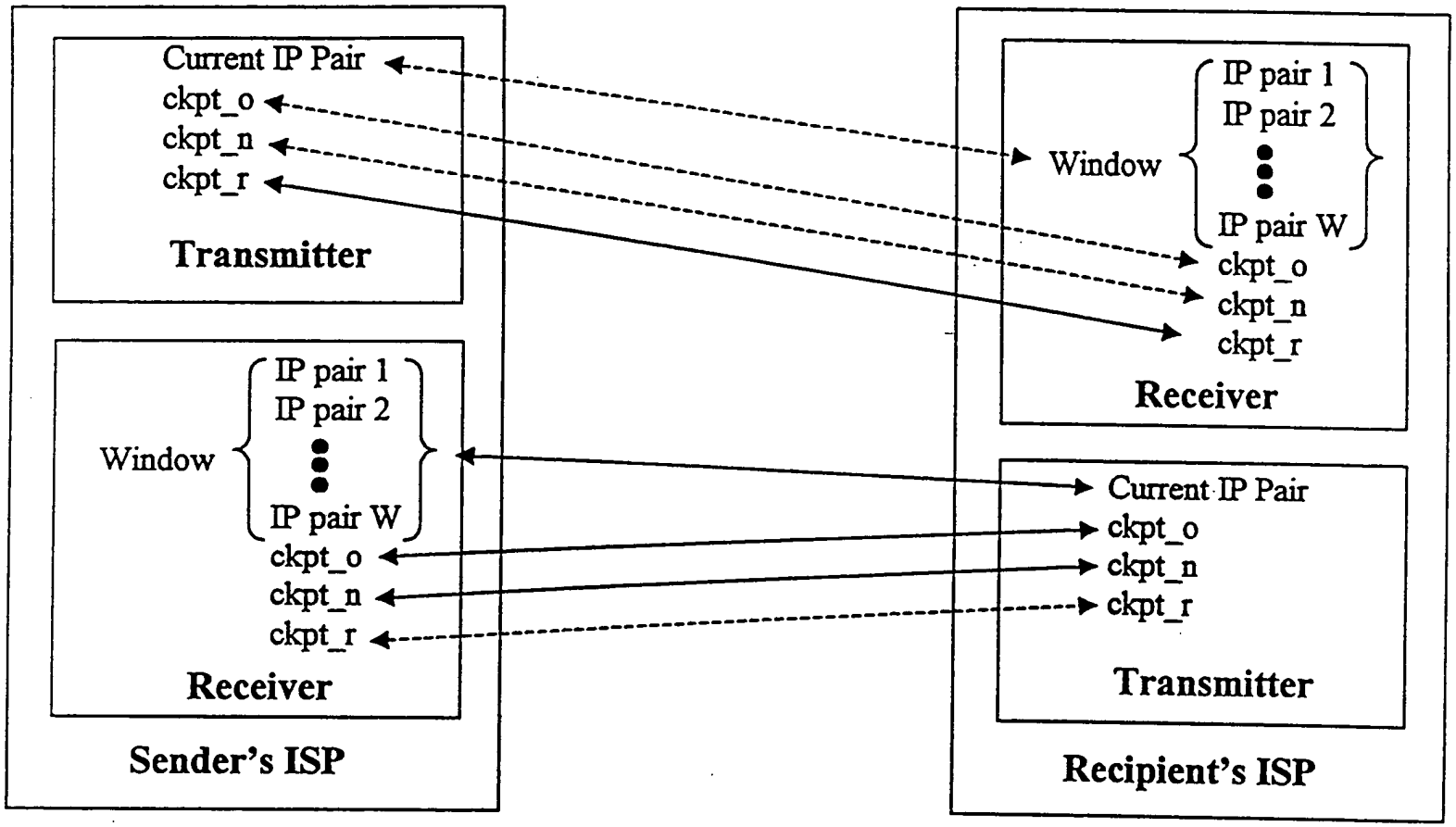


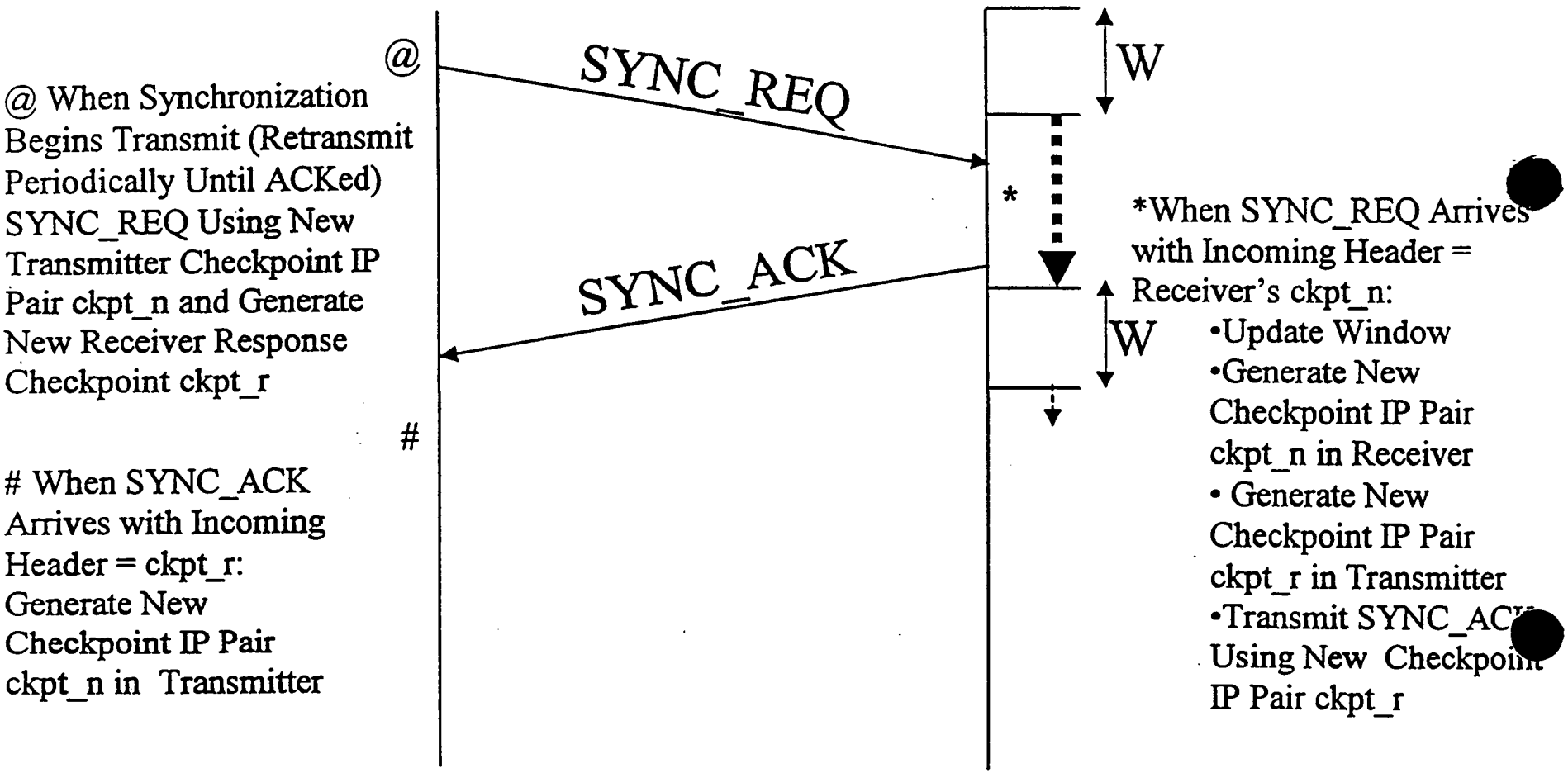
FIG. 13



Kept in Sync for Sender to Recipient Synchronizer

Kept in Sync for Recipient to Sender Synchronizer

FIG. 14



@ When Synchronization Begins Transmit (Retransmit Periodically Until ACKed) SYNC_REQ Using New Transmitter Checkpoint IP Pair ckpt_n and Generate New Receiver Response Checkpoint ckpt_r

When SYNC_ACK Arrives with Incoming Header = ckpt_r: Generate New Checkpoint IP Pair ckpt_n in Transmitter

*When SYNC_REQ Arrives with Incoming Header = Receiver's ckpt_n:
- •Update Window
- •Generate New Checkpoint IP Pair ckpt_n in Receiver
- • Generate New Checkpoint IP Pair ckpt_r in Transmitter
- •Transmit SYNC_ACK Using New Checkpoint IP Pair ckpt_r

FIG. 15

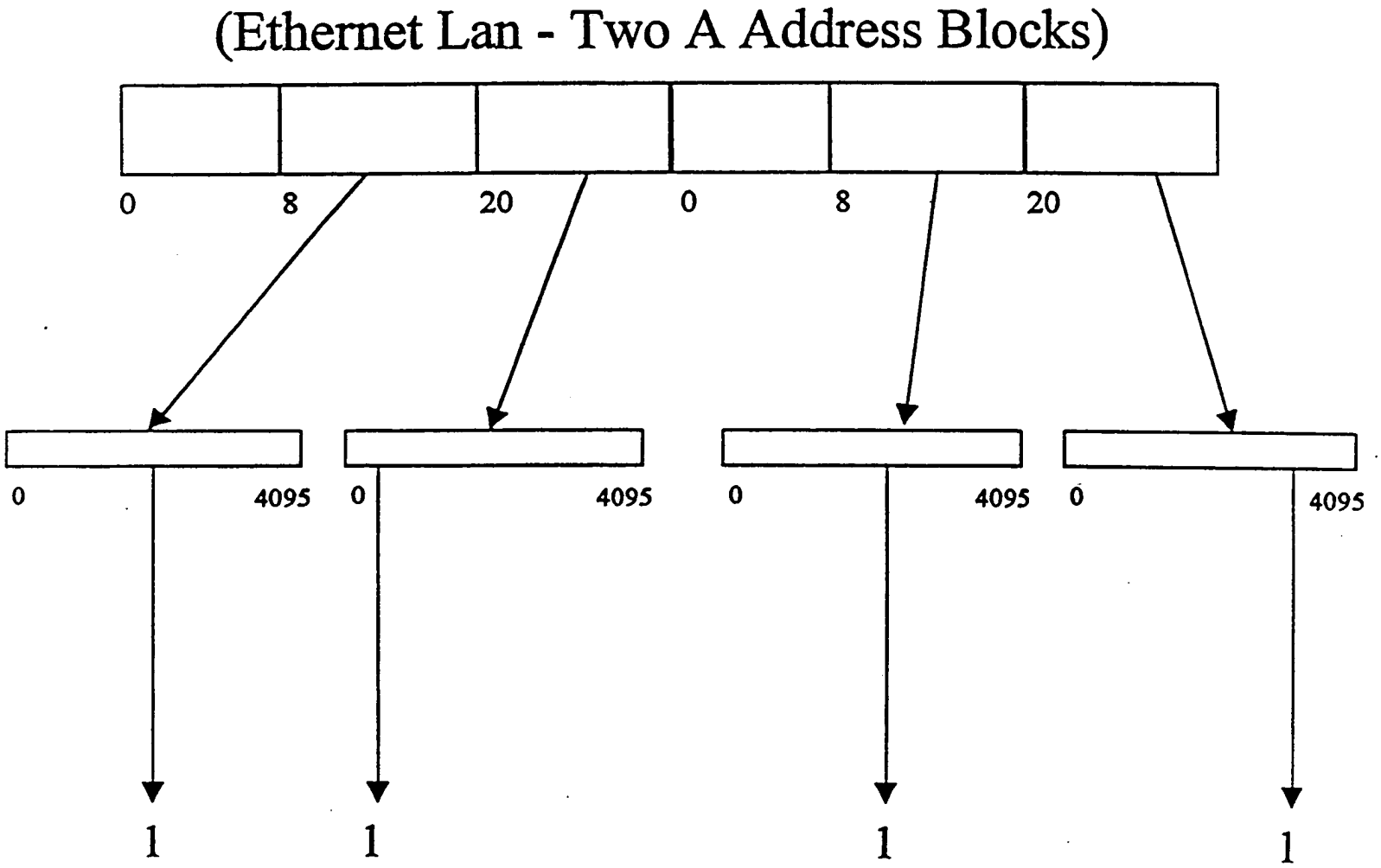

(Ethernet Lan - Two A Address Blocks)
0
8
20
0
8
20
0
4095
0
4095
0
4095
0
4095
1
1
1
1

FIG. 16

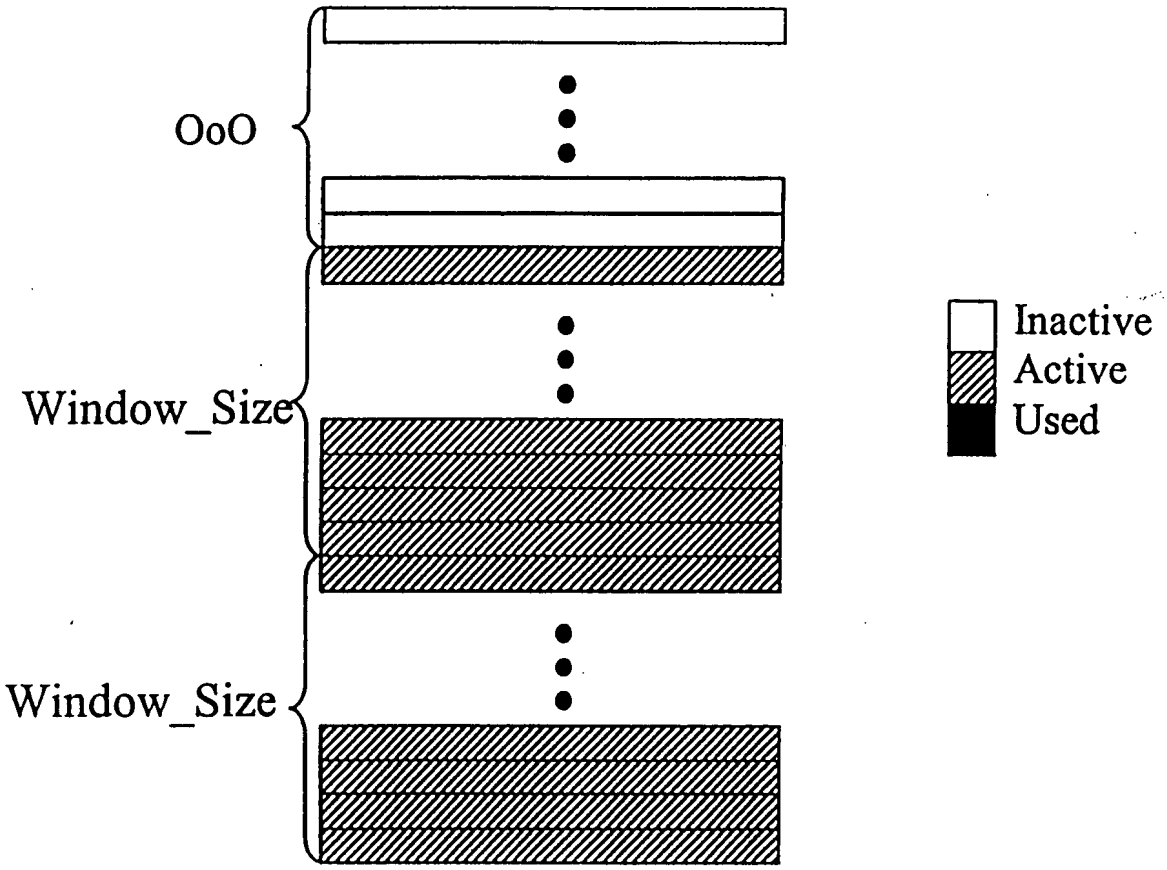


FIG. 17

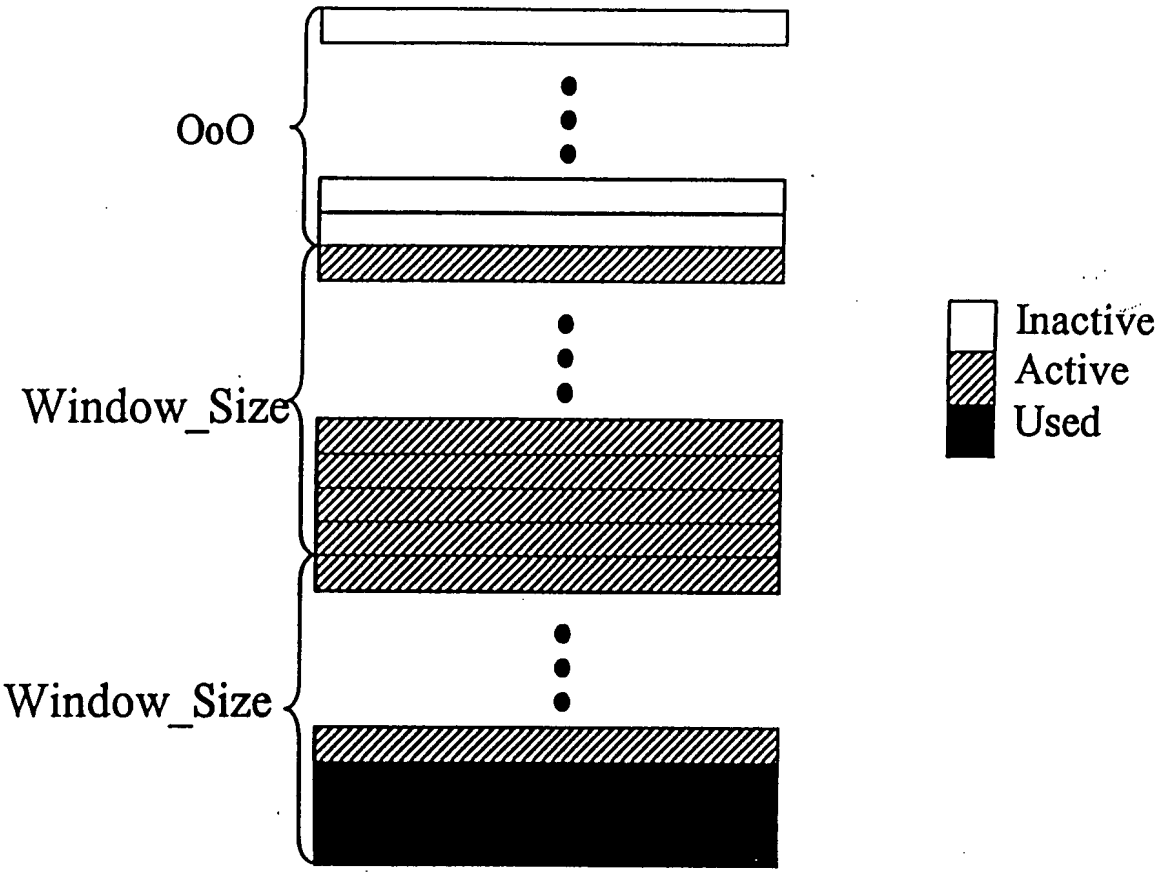


FIG. 18

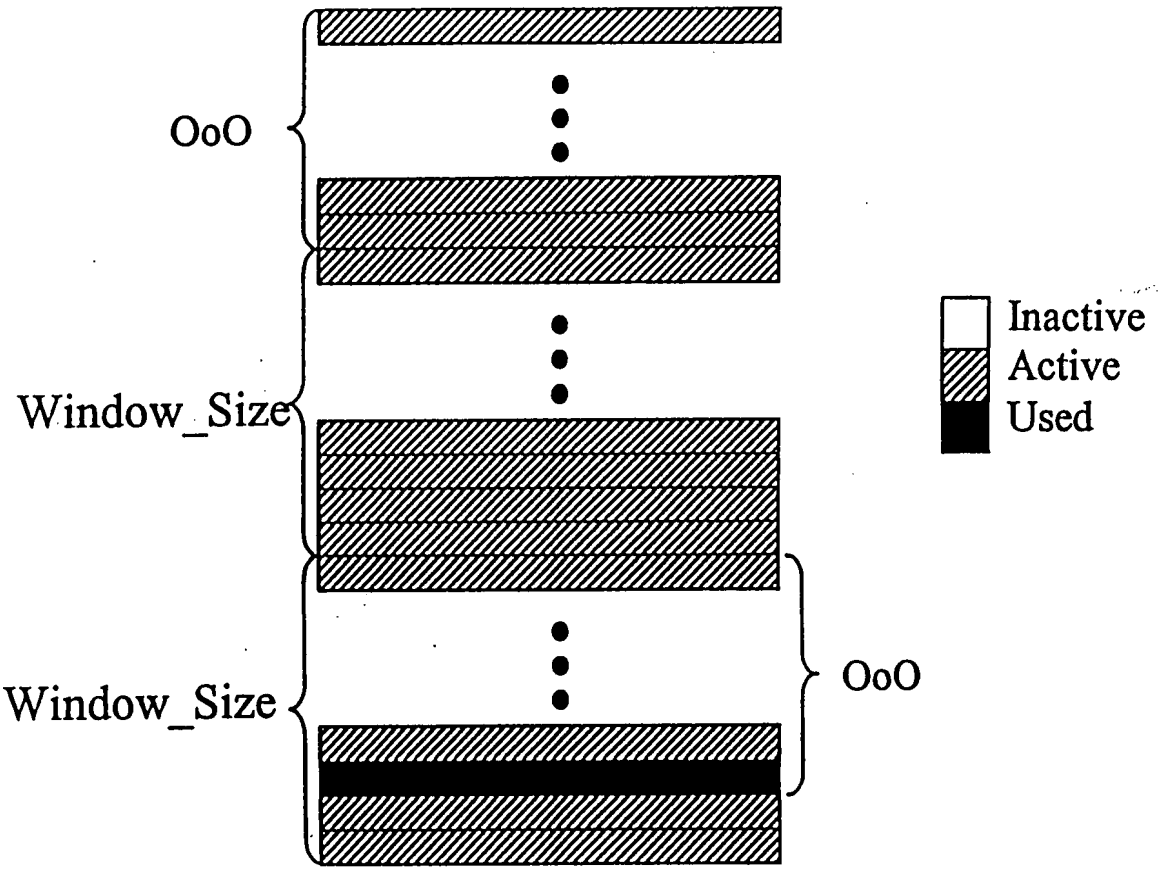


FIG. 19

FIG. 20

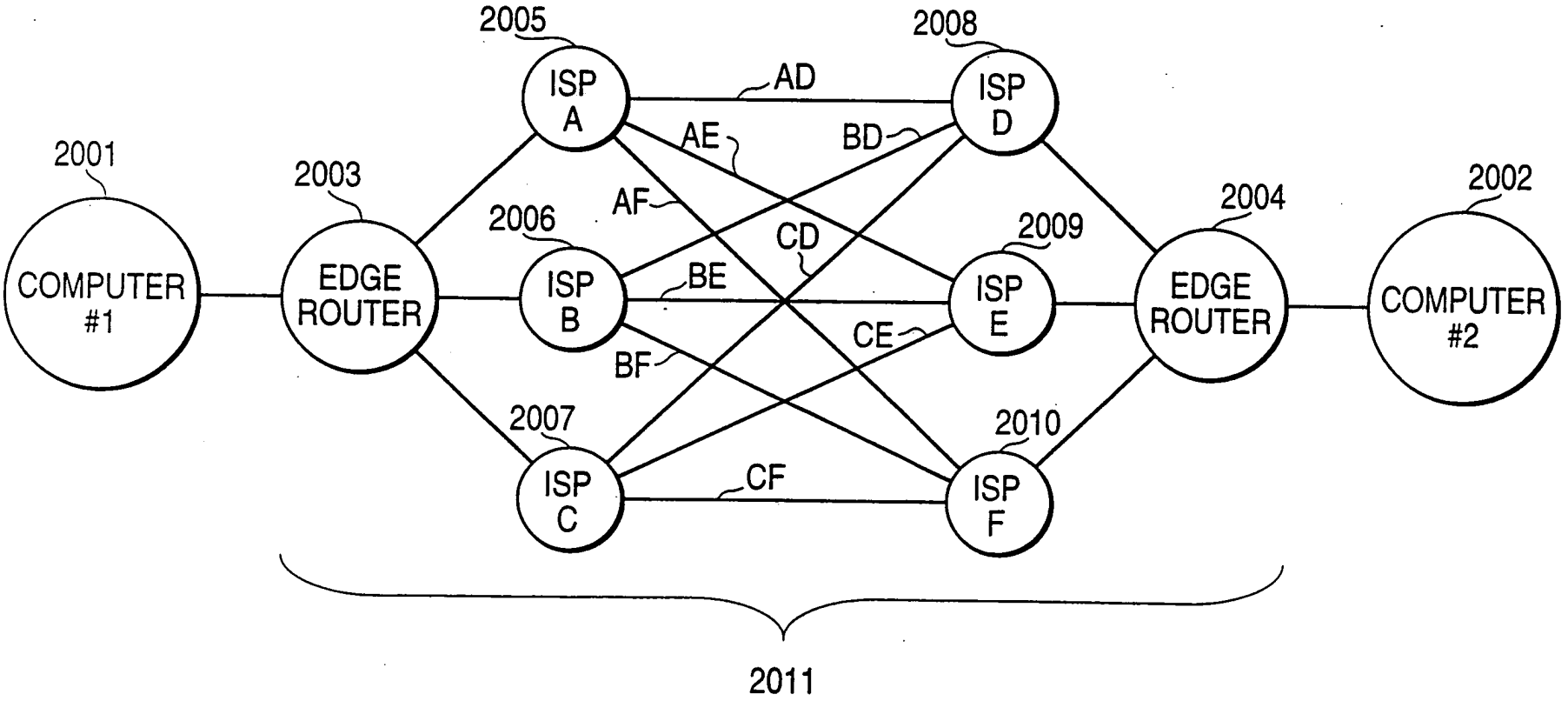
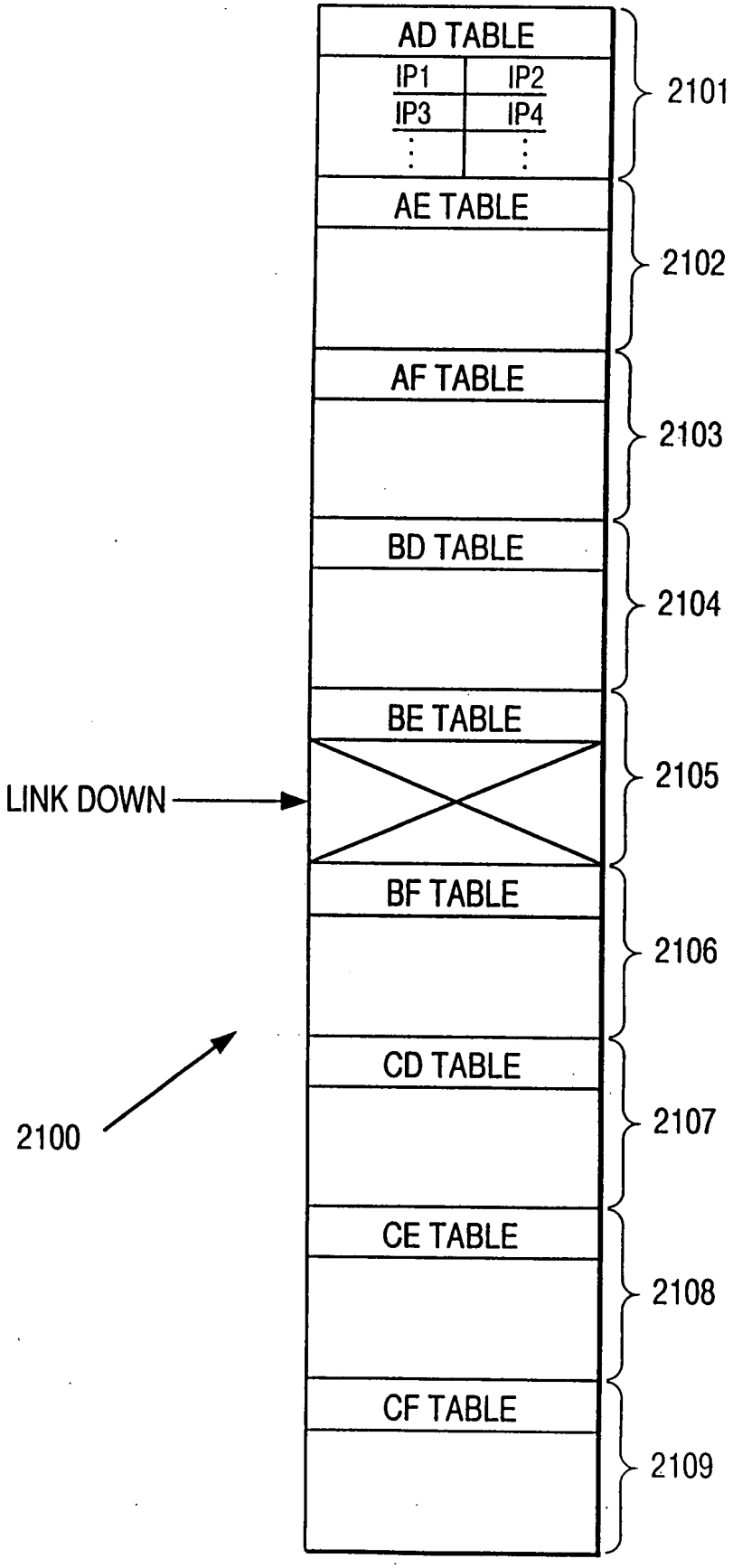

2005
2008
ISP A
ISP D
AD
AE
BD
2001
2003
AF
2006
CD
2009
2004
2002
COMPUTER #1
EDGE ROUTER
ISP B
BE
ISP E
EDGE ROUTER
COMPUTER #2
CE
BF
2007
2010
ISP C
CF
ISP F
2011

FIG. 21


AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
LINK DOWN
2105
BF TABLE
2106
CD TABLE
2107
2100
CE TABLE
2108
CF TABLE
2109

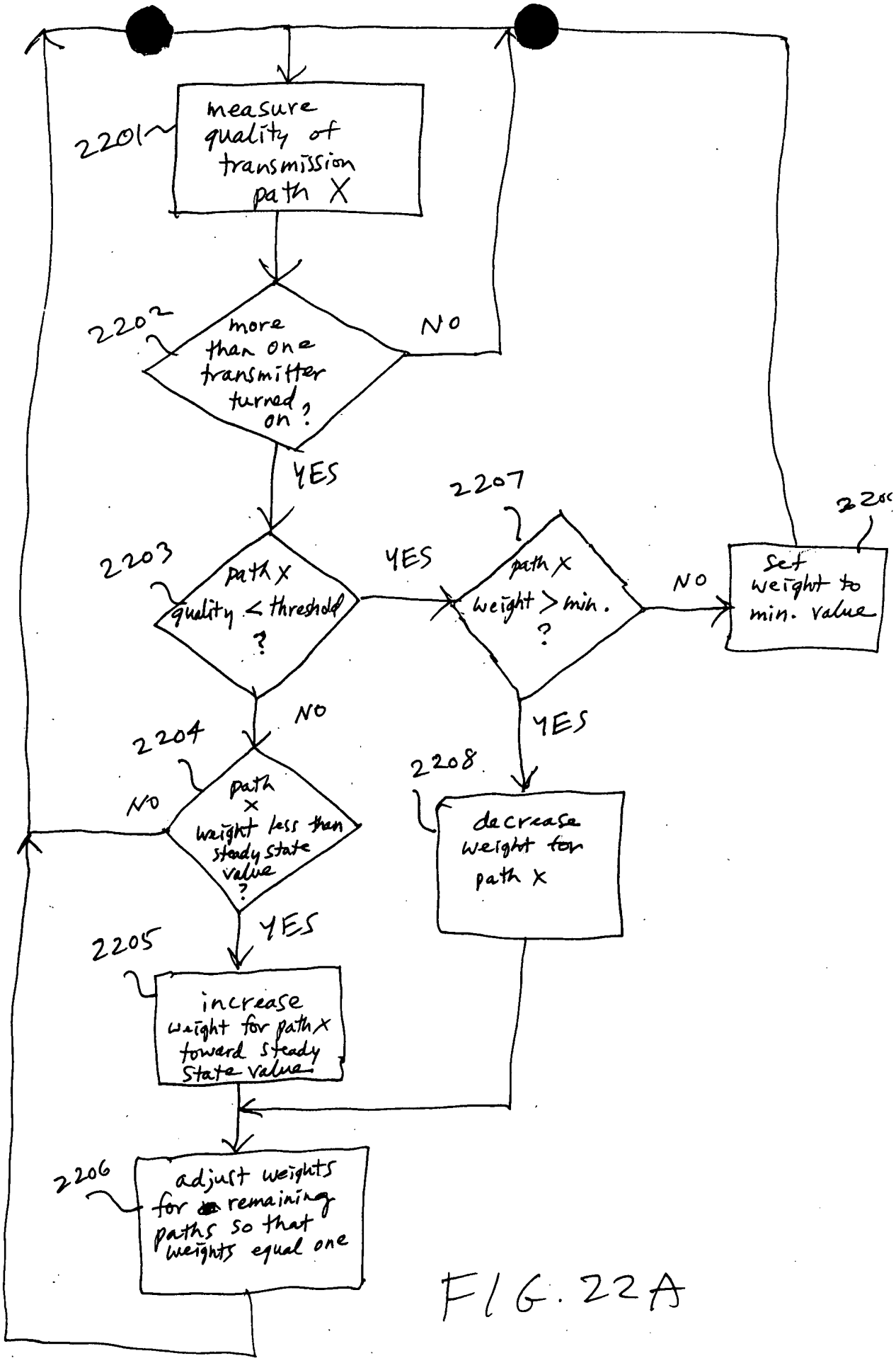

2201
measure quality of transmission path X
2202
more than one transmitter turned on?
NO
YES
2203
path X quality < threshold ?
YES
2207
path X weight > min. ?
NO
2208
Set weight to min. value
YES
NO
2204
path X weight less than steady state value ?
NO
YES
2205
increase weight for path X toward steady state value
2208
decrease weight for path X
2206
adjust weights for remaining paths so that weights equal one

FIG. 22A

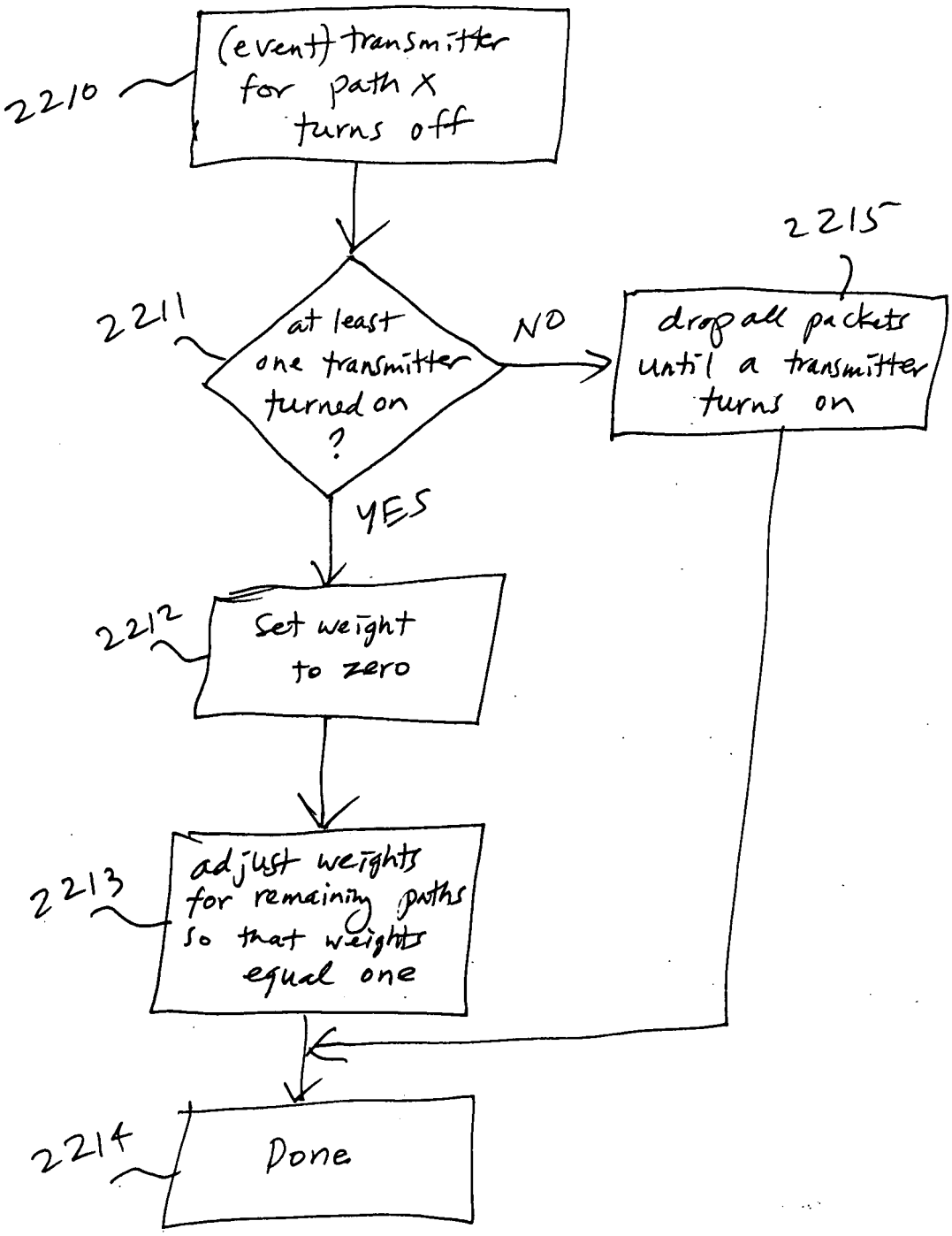

(event) transmitter for path X turns off
2210
at least one transmitter turned on ?
2211
NO
YES
drop all packets until a transmitter turns on
2215
Set weight to zero
2212
adjust weights for remaining paths so that weights equal one
2213
Done
2214

FIG. 22B

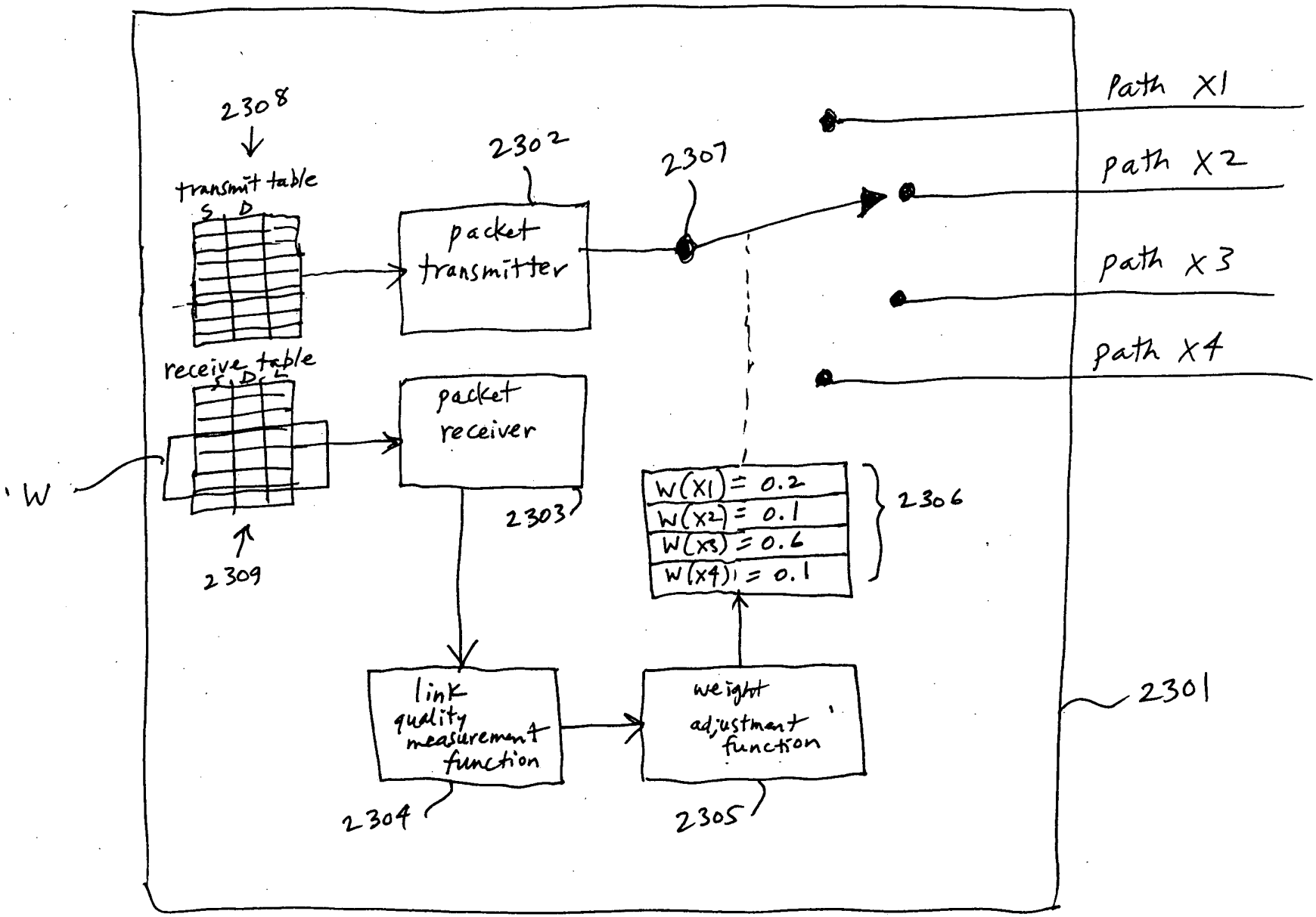


FIG. 23

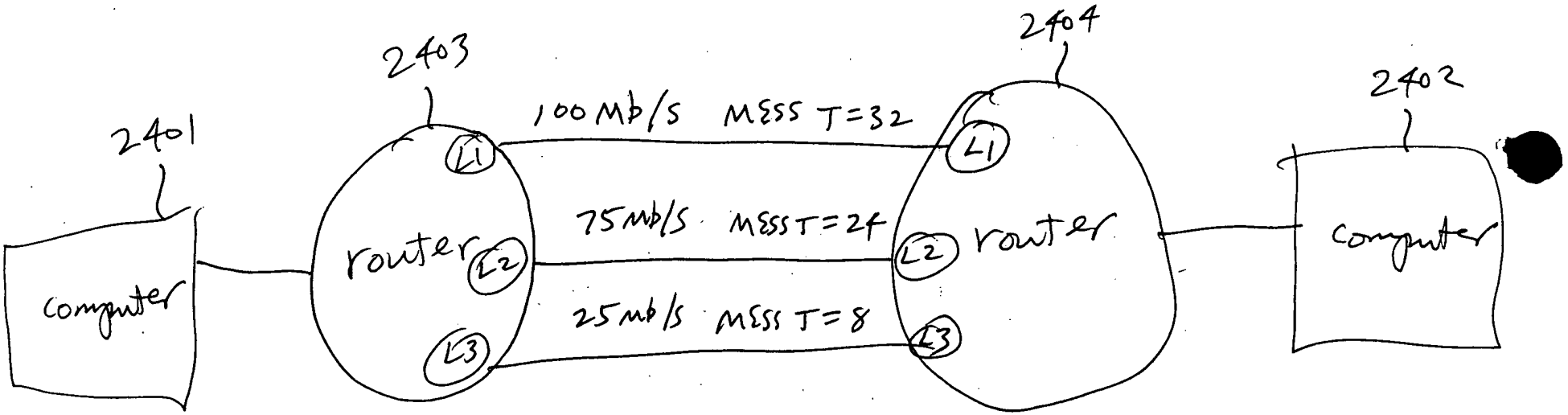

2401
2403
2404
2402
100 Mb/s MESS T=32
75 Mb/s MESS T=24
25 Mb/s MESS T=8
L1
L2
L3
L1
L2
L3
computer
router
router
computer

FIG. 24

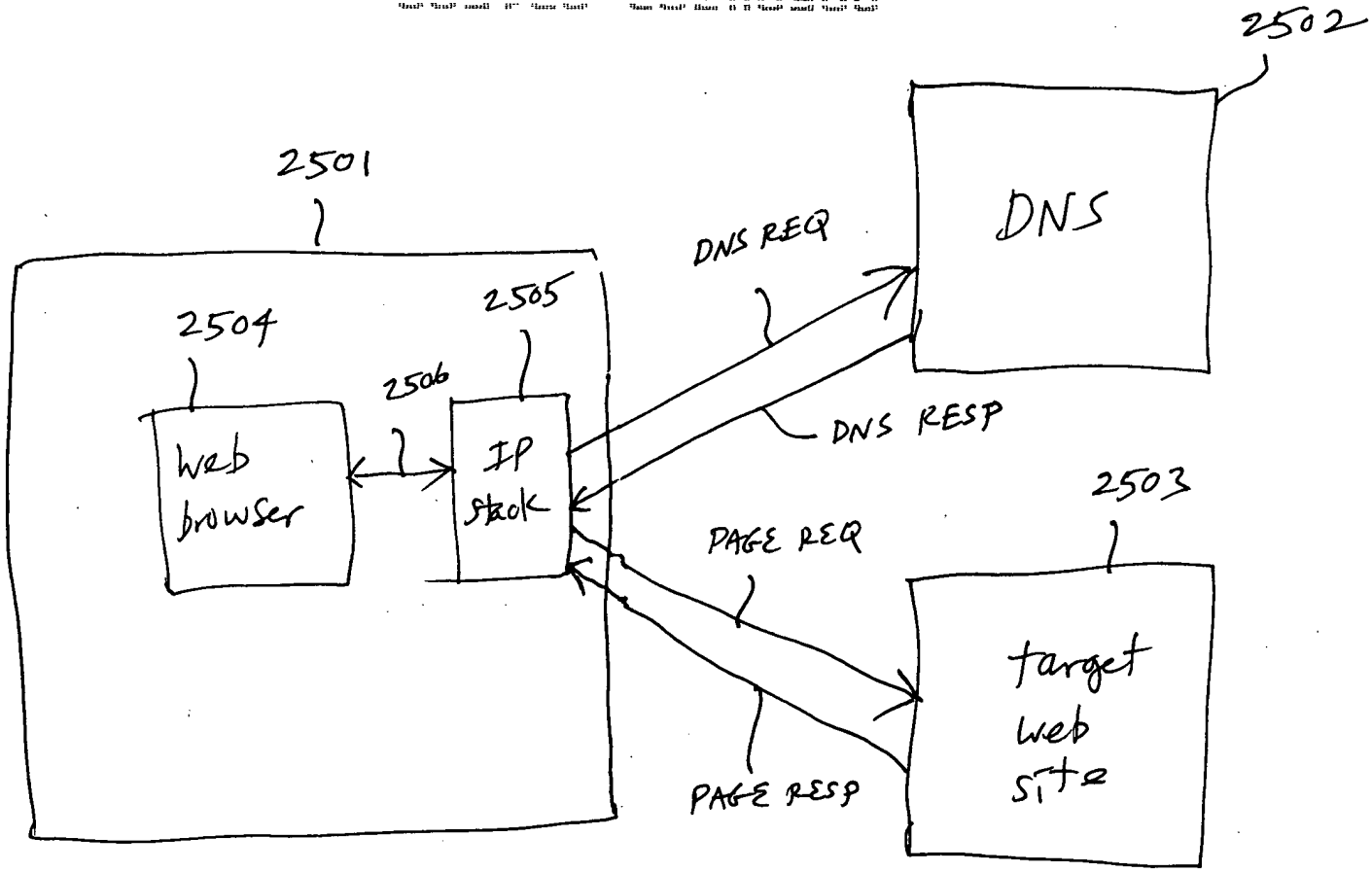


FIG. 25
(prior art)

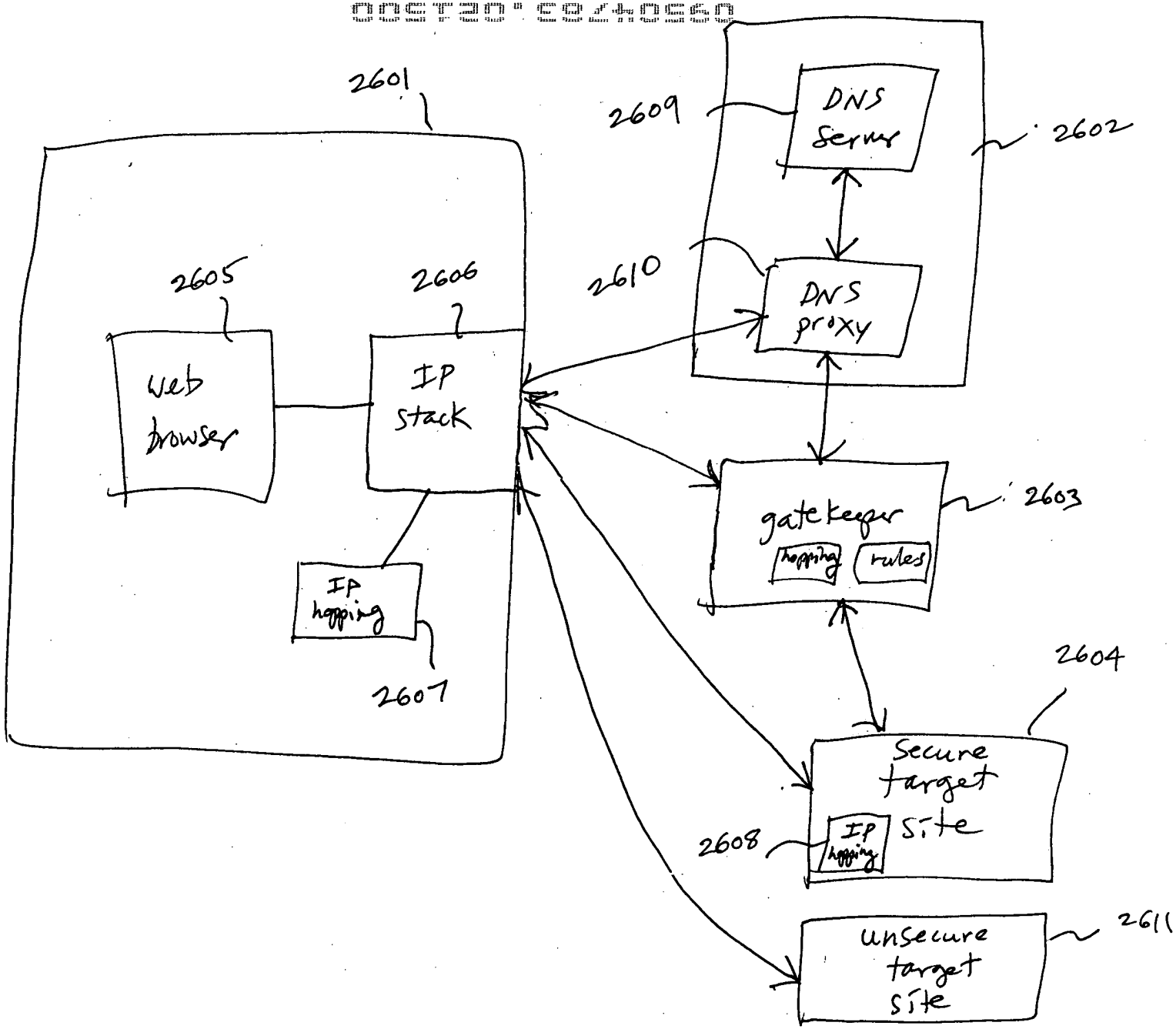

2601
2609
DNS Server
2602
2610
DNS proxy
2605
2606
Web browser
IP Stack
IP hopping
2607
gatekeeper
hopping
rules
2603
2604
Secure target site
IP hopping
2608
unsecure target site
2611

FIG. 26

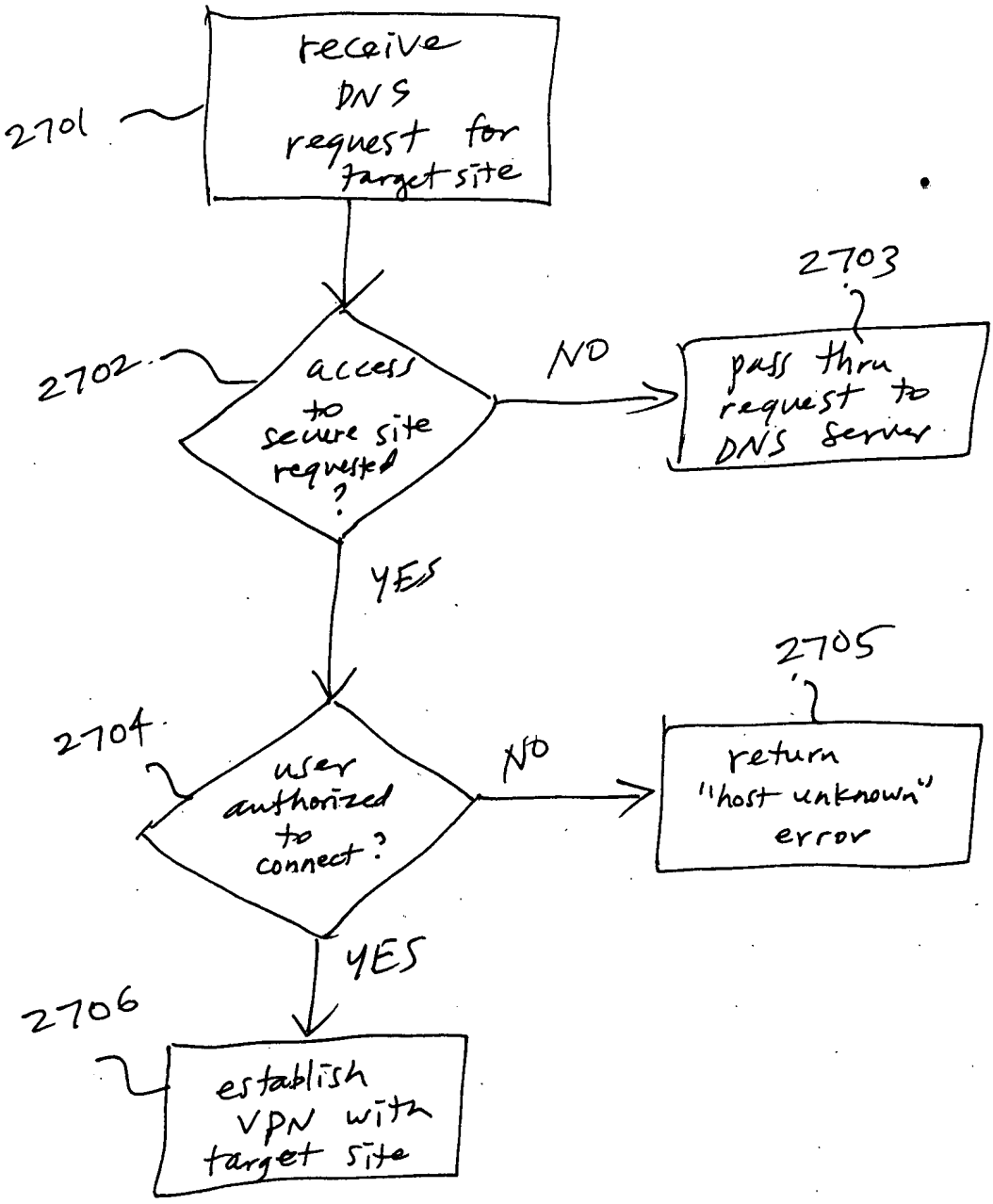

receive DNS request for target site
2701
access to secure site requested ?
2702
NO
2703
pass thru request to DNS server
YES
user authorized to connect ?
2704
NO
2705
return "host unknown" error
YES
2706
establish VPN with target site

FIG. 27

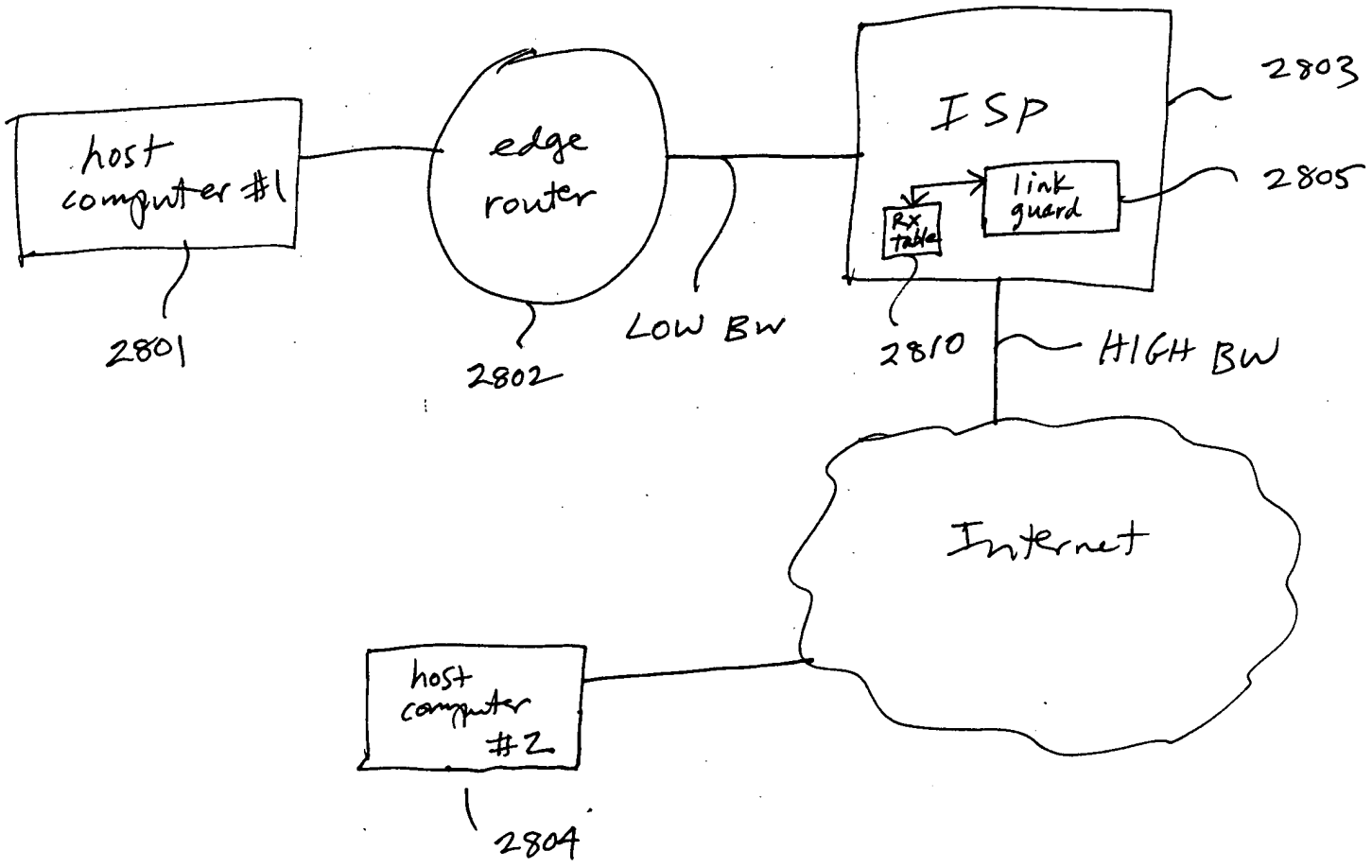

host computer #1
edge router
ISP
2803
link guard
2805
Rx table
2801
2802
LOW BW
2810
HIGH BW
Internet
host computer #2
2804

FIG. 28

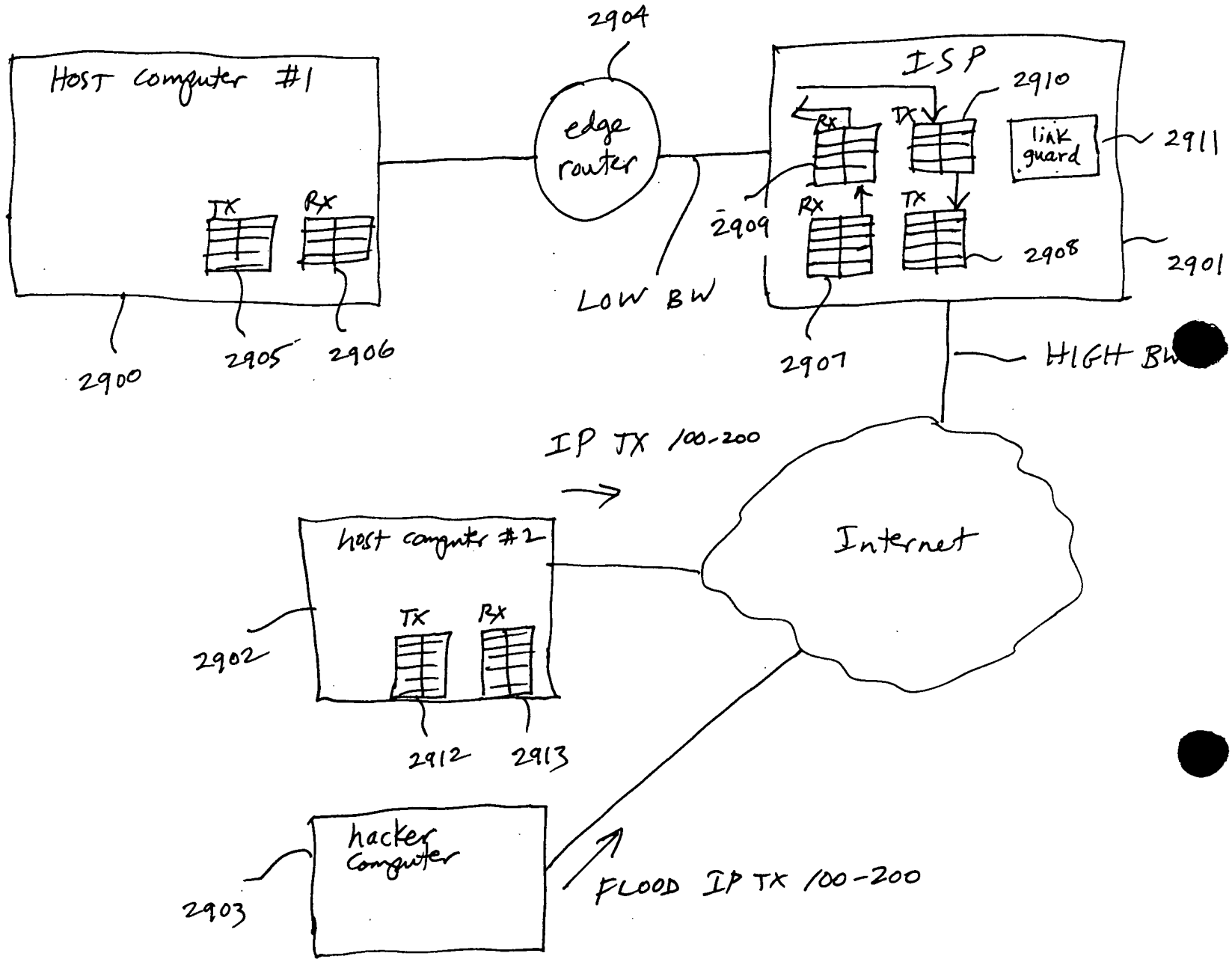


FIG. 29

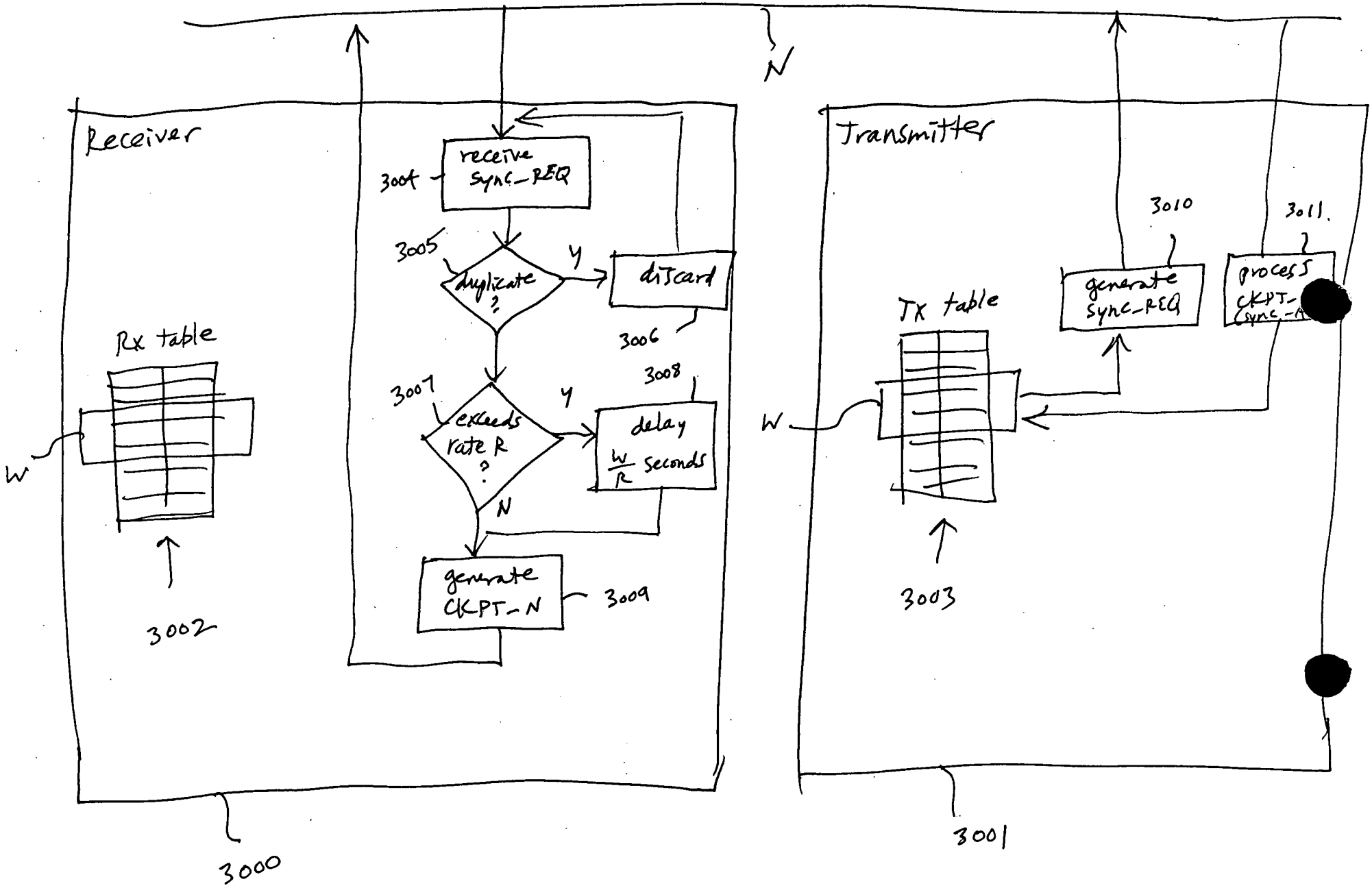

N
Receiver
Transmitter
receive sync_REQ
3004
3005
duplicate ?
Y
discard
3006
3007
exceeds rate R ?
Y
N
3008
delay W/R seconds
generate CKPT_N
3009
Rx table
W
3002
TX table
W
3003
3010
generate sync_REQ
3011
process CKPT-
3000
3001

FIG. 30

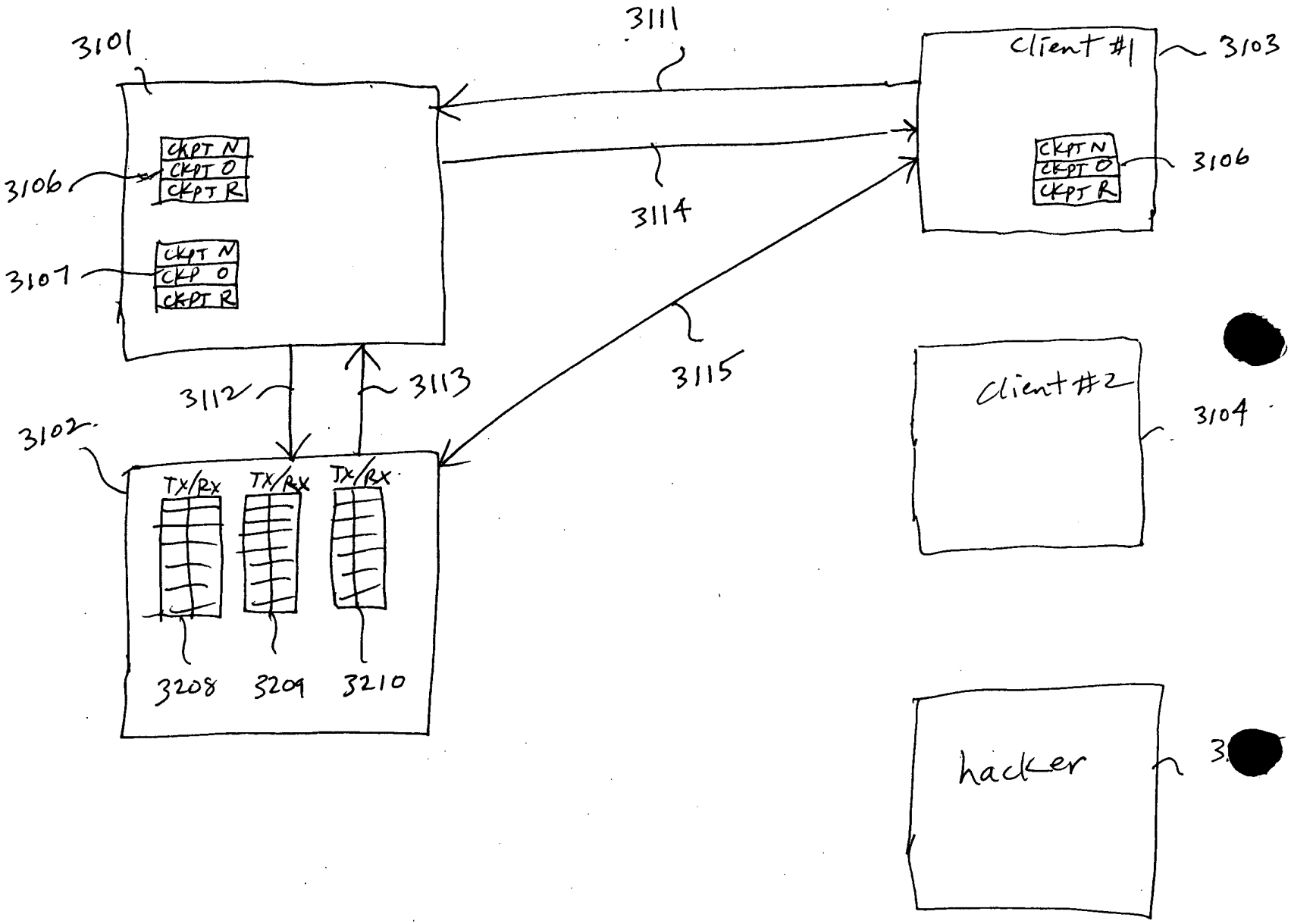

3101
3111
Client #1
3103
CKPT N
CKPT O
CKPT R
3106
3106
CKPT N
CKPT O
CKPT R
3107
3114
3112
3113
3115
Client #2
3104
3102
TX/RX
TX/RX
TX/RX
3208
3209
3210
hacker

FIG. 31

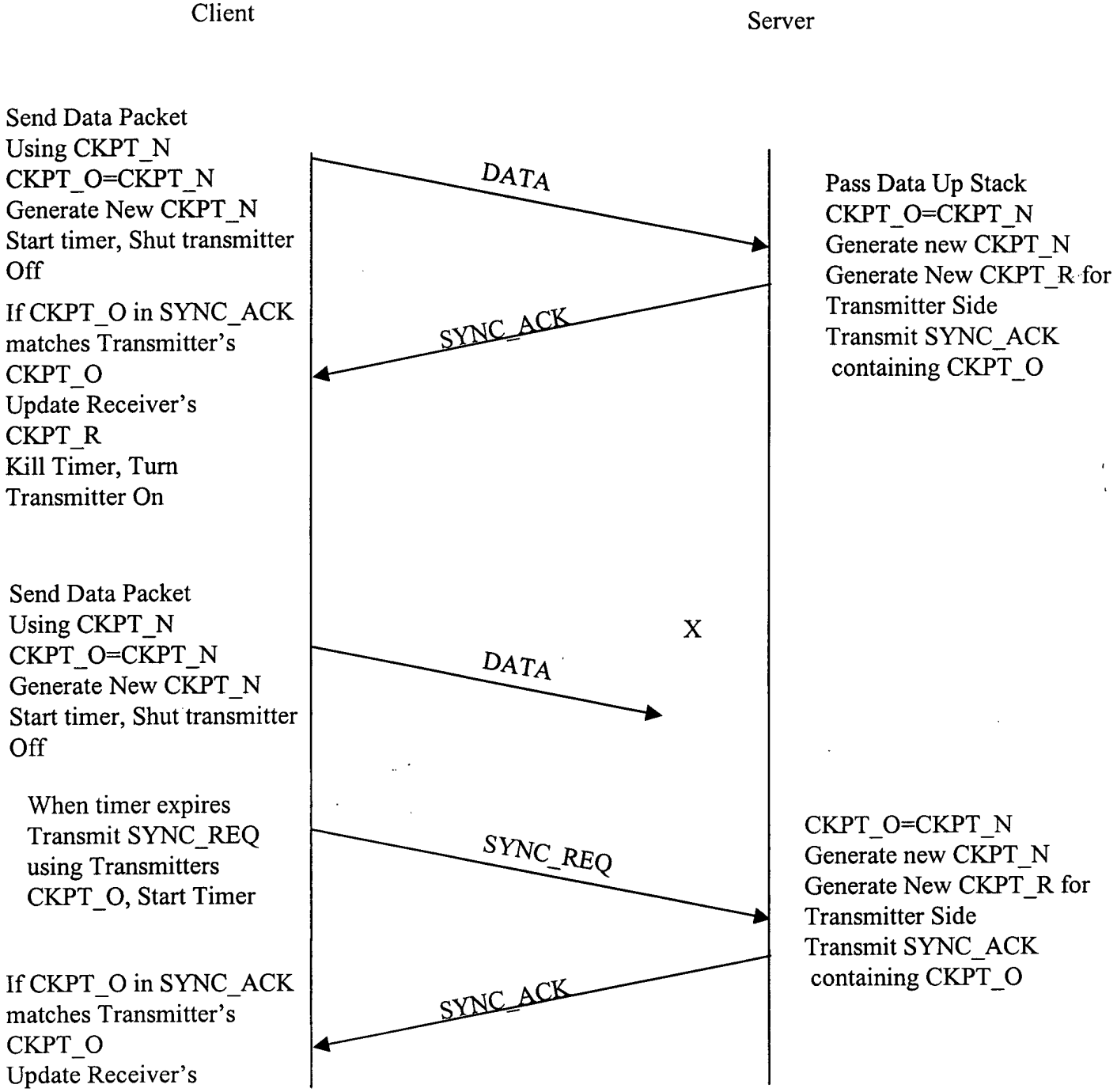

Client
Server
Send Data Packet
Using CKPT_N
CKPT_O=CKPT_N
Generate New CKPT_N
Start timer, Shut transmitter Off
DATA
Pass Data Up Stack
CKPT_O=CKPT_N
Generate new CKPT_N
Generate New CKPT_R for Transmitter Side
Transmit SYNC_ACK containing CKPT_O
SYNC_ACK
If CKPT_O in SYNC_ACK matches Transmitter's CKPT_O
Update Receiver's CKPT_R
Kill Timer, Turn Transmitter On
Send Data Packet
Using CKPT_N
CKPT_O=CKPT_N
Generate New CKPT_N
Start timer, Shut transmitter Off
X
DATA
When timer expires
Transmit SYNC_REQ using Transmitters CKPT_O, Start Timer
SYNC_REQ
CKPT_O=CKPT_N
Generate new CKPT_N
Generate New CKPT_R for Transmitter Side
Transmit SYNC_ACK containing CKPT_O
If CKPT_O in SYNC_ACK matches Transmitter's CKPT_O
Update Receiver's CKPT_R
Kill Timer, Turn Transmitter On
SYNC_ACK

FIG. 32